ECKAL-3.4

VAX11750 CACHE
TB DIAGNOSTIC

EP-ECKAL-DL-3.4
1 OF 2 APR 1985
COPYRIGHT© 1980-85

digital
MADE IN USA

ECKAL-3.4

VAX11750 CACHE
TB DIAGNOSTIC

EP-ECKAL-DL-3.4
2 OF 2 APR 1985
COPYRIGHT© 1980-85

digital
MADE IN USA

I D E N T I F I C A T I O N

PRODUCT ID: ZZ-ECKAL-3.4
PRODUCT TITLE: ECKAL32 VAX 11/750 CACHE/TRANSLATION BUFFER TEST
DECO/DEPO: 3.4
DATE: NOVEMBER 13, 1984
AUTHOR: DOMENIC ANDELLA, DAN MILLEVILLE
DEPARTMENT: BASE SYSTEMS DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1980, 1983
Digital Equipment Corporation, Maynard, Massachusetts 01754

This software is furnished under a license for use only on a single computer system and may be copied only with the inclusion of the above copyright notice. This software, or any other copies thereof, may not be provided or otherwise made available to any other person except for use on such system and to one who agrees to these license terms. Title to and ownership of the software shall at all times remain in DEC.

The information in this software is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

DEC assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DEC.

CONTENTS

1	ABSTRACT	2
2	REQUIREMENTS	2
2.1	HARDWARE	2
2.2	SOFTWARE REQUIREMENTS	2
2.3	PREREQUISITES	2
3	OPERATING INSTRUCTIONS	3
4	FUNCTIONAL DESCRIPTION	3
4.1	PROGRAM OVERVIEW	3
4.2	PROGRAM SIZE	5
4.3	PROGRAM RUN TIMES	5
4.4	RUN-TIME DYNAMICS	5
4.5	EVENT FLAGS	5
4.6	FAULT DETECTION	5
4.7	PERFORMANCE DURING HARDWARE FAILURES	5
4.8	SEQUENCE OF TESTING MULTIPLE UNITS	6
4.9	PROGRAM APPLICATIONS	6
4.10	USEFUL APT INFORMATION	6
4.11	TEST AND SUBTEST DESCRIPTIONS	7
4.11.1	TEST 1: CACHE CONTROL LOGIC TEST	9
4.11.2	TEST 2: NON-EXISTENT MEMORY REFERENCE TEST	10
4.11.3	TEST 3: CACHE SIMPLE ADDRESS PATHS TEST	10
4.11.4	TEST 4: CACHE SIMPLE DATA PATHS TESTS	11
4.11.5	TEST 5: CACHE HIT/MISS COMPARITORS TEST	12
4.11.6	TEST 6: CACHE TAG STORE DUAL ADDRESS TEST	12
4.11.7	TEST 7: CACHE DATA STORE DUAL ADDRESS TEST	14
4.11.8	TEST 8: CACHE BYTE/WORD WRITE MISS TEST	14
4.11.9	TEST 9: CACHE TAG STORE DYNAMIC MEMORY TEST	15
4.11.10	TEST 10: CACHE DATA STORE DYNAMIC MEMORY TEST	15
4.11.11	TEST 11: CACHE BYTE WRITE TEST	16
4.11.12	TEST 12: CACHE I/O SPACE REFERENCE TEST	16
4.11.13	TEST 13: NON CPU MEMORY REFERENCE TEST	17
4.11.14	TEST 14: CACHE LOCKED READ TEST	18
4.11.15	TEST 15: TB CONTROL LOGIC TEST	19
4.11.16	TEST 16: TB ADDRESS PATH TAG MEMORY TEST GROUP 0	20
4.11.17	TEST 17: TB ADDRESS PATH TAG MEMORY TEST GROUP 1	21
4.11.18	TEST 18: TB DATA PATH DATA MEMORY TEST GROUP 0	21
4.11.19	TEST 19: TB DATA PATH DATA MEMORY TEST GROUP 1	21
4.11.20	TEST 20: TB HIT/MISS COMPARITORS TEST GROUP 0	22
4.11.21	TEST 21: TB HIT/MISS COMPARITORS TEST GROUP 1	22
4.11.22	TEST 22: TB DUAL ADDRESS TAG STORE TEST GROUP 0	22
4.11.23	TEST 23: TB DUAL ADDRESS TAG STORE TEST GROUP 1	23
4.11.24	TEST 24: TB DUAL ADDRESS DATA STORE TEST GROUP 0	24
4.11.25	TEST 25: TB DUAL ADDRESS DATA STORE TEST GROUP 1	24
4.11.26	TEST 26: TB DYNAMIC TEST TAG STORE MEMORY GROUP 0	25

4.11.27 Test 27: TB DYNAMIC TEST TAG STORE MEMORY GROUP 1 25
4.11.28 TEST 28: TB DYNAMIC TEST DATA STORE MEMORY GROUP 0 26
4.11.29 TEST 29: TB DYNAMIC TEST DATA STORE MEMORY GROUP 1 26
4.11.30 TEST 30: TB DYNAMIC TEST OF VALID BIT GROUP 0 27
4.11.31 TEST 31: TB DYNAMIC TEST OF VALID BIT GROUP 1 27
4.11.32 TEST 32: TB RANDOM GROUP REPLACEMENT TEST . 28
4.11.33 TEST 33: TB MAPPING TEST 29
4.11.34 TEST 34: TB MODIFIED BIT TEST 30
4.11.35 TEST 35: TB CROSS PAGE REFERENCE TEST . . . 30
5 MAINTENANCE HISTORY 31

1 ABSTRACT

This program provides quick verification of the Cache and Translation Buffer(TB) portion of a VAX 11/750 CPU. Hardware errors are isolated to the function, with a halt occuring on error. The program is macro level four, that operates stand alone without the use of the Diagnostic Supervisor. If PCS is present, then PCS microcode will be loaded by this program.

2 REQUIREMENTS

2.1 HARDWARE

VAX 11/750 CPU

256 KB of memory

NOTE: The following range of memory addresses is assumed to be non existent due to the physical limitation of the number of memory array modules that can be inserted into the backplane. (8 array modules)

address 200000 through address EFF000

Console Terminal

Suitable load device such as TU58 or Automated Product Test (APT).

Unibus Excerciser Terminator (UET) is required for Cache Test 12, and is necessary for generating the SYNC POINT for scope looping.

2.2 SOFTWARE REQUIREMENTS

None

2.3 PREREQUISITES

The EVKAA VAX Hardcore Instruction Test must have been run successfully prior to the execution of the program.

3 OPERATING INSTRUCTIONS

1. Insert the TU58 Cassette on which ECKAL.EXE resides into the console drive.

2. Insure that the Restart/Halt switch is set to Halt.

3. Load the program normally by typing:

   >>> B DDA0 <CR>

   This will load and start the program in the first good 64K partition of memory + 200(X).

   NOTE: The program is NON-PIC; if the first good 64K partition is not PA0 - 64K, the program will behave unpredictably.

If the program is loaded by means other than booting, such as RD, it should be loaded at address 0, and started at address 200.

Every ten passes of the program an Endpass routine outputs the message 'ECKAL x.x DONE!" to the console and rings the bell.

4 FUNCTIONAL DESCRIPTION

4.1 PROGRAM OVERVIEW

The program consists of 35 tests (14 Cache tests and 21 TB tests). They are as follows:

1. TEST 1: CACHE CONTROL LOGIC TEST
2. TEST 2: CACHE NON-EXISTENT MEMORY REFERENCE TEST
3. TEST 3: CACHE SIMPLE ADDRESS PATHS TEST
4. TEST 4: CACHE SIMPLE DATA PATHS TEST
5. TEST 5: CACHE HIT/MISS COMPARITORS TEST
6. TEST 6: CACHE TAG STORE DUAL ADDRESS TEST
7. TEST 7: CACHE DATA STORE DUAL ADDRESS TEST
8. TEST 8: CACHE BYTE/WORD WRITE MISS TEST
9. TEST 9: CACHE TAG STORE DYNAMIC MEMORY TEST
10. TEST 10: CACHE DATA STORE DYNAMIC MEMORY TEST

11. TEST 11: CACHE BYTE WRITE TEST
12. TEST 12: CACHE I/O SPACE REFERENCE TEST
13. TEST 13: CACHE NON-CPU MEMORY REFERENCE TEST
14. TEST 14: CACHE LOCKED READ TEST
15. TEST 15: TB CONTROL LOGIC TEST
16. TEST 16: TB ADDRESS PATH TAG MEMORY TEST GROUP 0
17. TEST 17: TB ADDRESS PATH TAG MEMORY TEST GROUP 1
18. TEST 18: TB DATA PATH DATA MEMORY TEST GROUP 0
19. TEST 19: TB DATA PATH DATA MEMORY TEST GROUP 1
20. TEST 20: TB HIT/MISS COMPARITORS TEST GROUP 0
21. TEST 21: TB HIT/MISS COMPARITORS TEST GROUP 1
22. TEST 22: TB DUAL ADDRESS TAG STORE TEST GROUP 0
23. TEST 23: TB DUAL ADDRESS TAG STORE TEST GROUP 1
24. TEST 24: TB DUAL ADDRESS DATA STORE TEST GROUP 0
25. TEST 25: TB DUAL ADDRESS DATA STORE TEST GROUP 1
26. TEST 26: TB DYNAMIC TEST TAG STORE MEMORY GROUP 0
27. TEST 27: TB DYNAMIC TEST TAG STORE MEMORY GROUP 1
28. TEST 28: TB DYNAMIC TEST DATA STORE MEMORY GROUP 0
29. TEST 29: TB DYNAMIC TEST DATA STORE MEMORY GROUP 1
30. TEST 30: TB DYNAMIC TEST OF VALID BIT GROUP 0
31. TEST 31: TB DYNAMIC TEST OF VALID BIT GROUP 1
32. TEST 32: TB RANDOM GROUP REPLACEMENT TEST
33. TEST 33: TB MAPPING TEST
34. TEST 34: TB MODIFIED BIT TEST
35. TEST 35: TB CROSS PAGE REFERENCE TEST

4.2 PROGRAM SIZE

22 kilobytes

4.3 PROGRAM RUN TIMES

Ten complete physical passes of the program take less than ten seconds.

4.4 RUN TIME DYNAMICS

None

4.5 EVENT FLAGS

Not Applicable

4.6 FAULT DETECTION

The hardware error detecting mechanism is compare and branch. The reporting mechanism is HALT.

Cache and Translation Buffer Tag and Data parity errors will cause a Machine Check Exception to occur. Error information for this type of error is contained in the Machine Check Logout Stack. The format of this stack is shown in "ECKAL1.LIS".

Errors that do not result in a Machine Check Exception, Write NXM Interrupt, or Access Control Violation Interrupt will cause a halt to occur in the respective test. Useful error information is contained in the comments after each Error halt.

Due to the inability to force parity errors at the Macro level, one-hundred percent error detectinn cannot be guaranteed. However, ninety-seven percent of the stuck-at-zero(S-A-0) and stuck-at-one(S-A-1) faults should be detected.

4.7 PERFORMANCE DURING HARDWARE FAILURES

On the occurence of an unexpected exception or interrupt, the program w' l halt in the Trap Catcher Block(TCB).

On a power fa'lure the program should be restarted.

4.8 SEQUENCE OF TESTING MULTIPLE UNITS

Not Applicable

4.9 PROGRAM APPLICATIONS

MANUFACTURING

Manufacturing can use the program for gate array and module quick-verification. It may also prove useful for environmental (thermal) testing.

FIELD SERVICE

Field Service can use this program for verification of customer installation, fault isolation, repair, repair verification, preventive maintenance, and verification of Engineering Change Order (ECO) installation.

CUSTOMER

The customer can use the program to provide verification of self-installation, preventive maintenance, and fault detection and isolation.

4.10 USEFUL APT INFORMATION

The following APT information is available to the user at the indicated locations.

APT__TEST__NUM	FE50
APT__TEST__COUNT	FE52
APT__PASS__CNT	FE54

APT__TEST__NUM

An absolute test number is located at this word address. This location is updated at the beginning of every test.

APT__TEST__COUNT

This word address contains the incremented test count. It is incremented at the beginning of each test and compared with the Absolute Test Number.

APT__PASS__CNT

This location contains the number of successful passes. completed.

4.11 TEST AND SUBTEST DESCRIPTIONS

PROGRAM EXECUTION SEQUENCER

A test sequence check is made at the beginning of each test. This check is necessary to prohibit the program from erroneously jumping into some other test and yielding misleading results. An Absolute Test Number, known as APT_TEST_NUM" and located in Physical Word Address FE50, is compared with the contents of a counter, known as "APT_TEST_COUNT" and located in Physical Word Address FE52. The Absolute Test Number is loaded at the beginning of each test, and then the counter is incremented.

If a Sequence Error Halt occurs, the contents of word location FE50 would indicate to what test number, in Hex, the test had mistakenly jumped. Subtracting 1 from the contents of word location FE52 would indicate the test number where the erronious jump occured.

Remember that the listing refers to Absolute Test Numbers 1-35 decimal, while APT_TEST_NUM, AND APT_TEST_COUNT VALUES will be Hex.

SCOPE LOOPS

Scope loops are provided for each test. After the error halt, an unconditional Branch instruction resides with a displacement back to the beginning of the subtest or other useful point. The user may then replace the HALT with a NOP and continue the program for a scope loop. Thus the user may implement a loop on error by only two steps:

1. Deposit 01 in place of 00 at PC of Halt

2. Type C to continue the program

Tnis same type of looping mechanism is provided in the service routines for Machine Checks, Write Non-Existent Memory, and Access Control Violations.

SYNC POINT

To aid in scope looping Cache tests that have a Cache Flush, a SYNC POINT is available at slot 4, pin C90 of the UBI module,(UBUS MSYN L). This signal is generated each time by reading the UET Address Register.

UNEXPECTED EXCEPTIONS AND INTERRUPTS

The System Control Block(SCB) is set up to point to Halts in a Trap Catcher Block(TCB). Therefore, if the program receives any unexpected interrupts or exceptions, other than Machine Checks, Write Non-Existent Memory, of Access Control Violations(ACV), it will halt in a well defined place which identifies the type of trap.

Service routines only exist for the three type of traps aforementioned.

INITIALIZATION ROUTINE

This routine initializes the pass count, sets up the Stack Pointers, loads the SCB vectors to point to Halts in the Trap Catcher Block. The three service routine vectors are written, all flags are cleared (APT_ERR, NXM_WT, NXM_RD, ACCVIO). The IPL is lowered to zero, and the stack is pushed in order to insure that the current mode is Kernel and is executing on the Kernel Stack.

GENERAL CACHE TEST NOTES

The first four Cache tests will be implemented in a straight line manner (no loops) to eliminate the possibility of getting Cache hits on instruction references. Hardware faults might cause the program to behave unpredictably, leaving the operator with little or no indication of what the fault was.

A Cache flush (invalidation of all Cache locations) is performed each time the Cache is enabled, clearing bit<0> of the Cache Disable Register (CADR).

The Cache flush is used extensively throughout all Cache tests to initialize conditions at the beginning of each test, and in some cases, to eliminate the possibility of the test code causing hits. This would be as a result of looping in tests utilizing DO LOOPS.

When making Physical Address references(PAD<23:00>, bits<1:0> refer to one of four bytes in the Cache Data block, bits<11:02> are the Index into 1024 Cache Tag and Data Store locations, and bits <23:12> comprise the Cache Tag Store.

Three techniques are used in protecting Cache Index Locations under test: the Cache Flush, Longword Storage Directive, or

Conditional Assembly Directives. There use is test dependent. The Cache Flush is explained above. A Longword Storage Directive used at the beginning of the test will reserve a location which can be manipulated to form a unique Cache Index location, as long as the test is less than 1000 bytes. (All tests are well under 1000). By clearing bits<23:12> of this reserved location, the Index value bits<11:02> can be used for any desired Tag Store Address.
The third method makes use of Conditional Assembly Block Directives, in most cases to protect more than one Index location for a particular test. By storing the current Index location in a location called TEMP, various testing statements can be performed, forcing the Assembler not to use any Test Index locations.

Certain routines and Cache Tests will cause assembly errors if their lengths are exceeded. This is necessary for proper operation of the assembly block directives used to protect certain Cache Index locations under test. The routines and Cache tests that will cause the assembly errors are the Write NXM Interrupt Service Routine, and Cache tests 1, 5-13.

4.11.1 TEST 1: CACHE CONTROL LOGIC TEST -

Subtest 1 first checks for the presence of a Unibus Exerciser Terminator (UET) which is necessary to generate scope Sync Points and is used in Cache Test 13. The subtest then goes on to enable the Cache and check that the Cache Error Register (CAER) is cleared.

Subtest 2 makes a write reference to PA zero (Cache Index loc. zero). Since a Cache flush was performed first, the referenced location shoud have been invalidated causing the write to result in a Cache miss. The key hardware checked here includes the Cache Tag Store, Equals Checker (Hit/Miss comparitors), and their respective I/O paths.

Subtest 3 first writes zeroes into PA zero, then makes a read reference to PA zero which should result in a Read hit. The data read from the Cache Data Store is checked for zero. All the I/O paths throughout the Cache are thus checked for being cleared (no bits stuck high).

Subtest 4 writes zeroes into PA 03F000 and checks that a miss occurs. This allows as many bits to be set as possible in the Cache Tag Store path (PAD<17:12>) without causing a Write NXM Interrupt to occur. The minumum memory configuration is 256 kilobytes.

Subtest 5 writes ones into PA 03F000, reads and checks for a

read hit, and verifies that the correct data is read. This test verifies that bits <17:12> in the Tag Store path and all bits in the Data Store path can be set.

4.11.2 TEST 2: NON-EXISTENT MEMORY REFERENCE TEST -

Since the minimum amount of memory required is 256 kilobytes, write references to Physical Address<23:18> may cause a Write NXM Interrupt to occur. The referenced Cache location however, will be cached. This test will verify this operation and that bits<23:18> in the Cache Tag Store path can be set. An NXM_WT_FLAG will be set prior to any write references that may be non-existent. In this manner, when an interrupt occurs the service routine will clear the flag and return from the interrupt. The flag is then tested to determine if, in fact, the interrupt actually occured. In this test a write reference to a Tag Store = EFF, and Index equal to that contained in label 'INDEX1' is attempted. This address is guaranteed NXM because of the physical limitation of the number of memory array modules that can be inserted into the backplane. With the full complement of eight array modules (each storing 256 kilobytes) the highest possibe memory address would be 1FFFFF. Writing to a Tag of EFF will cause a WT NXM Interrupt to occur, but the Cache Tag Store should have written EFF, and also the correct Data Store (in this test all A's). This technique will be used a number of times in order to test all of the Cache Tag Store.

Failures in this type of test might manifest themselves in the form of an unexpected Read NXM Machine check. If, for example, the write to NXM did not cache, when you went to read that location, you would get a miss, and then try to read the NXM address from main memory. This would result in a Read NXM Machine Check.

It should be noted nonetheless, that the Write NXM Interrupt can be blocked out by raising the IPL level to 1F, and indeed this is done in some of the Cache tests by necessity. Where possible, however, expected interrupts are not blocked in order that any other unexpected interrupt might occur as a result of a Cache or TB hardware error.

4.11.3 TEST 3: CACHE SIMPLE ADDRESS PATHS TEST -

This test checks the input and output paths of the Cache Tag Store rams for bits S-A-0, S-A-1, and shorted together by referencing an Index value specified by label INDEX2 , and

writing a minimal bit fault pattern set to the Tag Store and checking for hits.

Four patterns are used in this test, they are as follows:

Tag Store Test Pattern1 = AAA
Tag Store Test Pattern2 = 555
Tag Store Test Pattern3 = 333
Tag Store Test Pattern4 = F0F

The test is subdivided into four subtests, one for each pattern used. Each subtest will make a write reference to the respective pattern. Since all four patterns will address non-existent memory, a Write NXM Interrupt will occur. The NXM_WT_FLAG is tested after the write to insure that the expected interrupt did, in fact, occur. A read reference is then performed to that address and it is checked for a read hit.

Failures in this test would manifest themselves by either halting on error in the respective subtest, or causing an unexpected Read NXM Machine to occur. The Read NXM Machine Check would be the result of writing an incorrect address in the Tag Store.

4.11.4 TEST 4: CACHE SIMPLE DATA PATHS TESTS -

This test checks the input and output paths of the Cache Data Store rams and Output Driver for bits S-A-0, and S-A-1, and shorted together by referencing an Index location specified by the label 'INDEX3' and writing a minimal bit fault pattern set across Cache<31:00>, and checking for correct data.

The test is divided into six subtests, each of which writes a test pattern into PA zero, reads PA zero, and then checks for the correct data results.

The six test patterns used are as follows:

Data Test pattern1 = AAAAAAAA
Data Test pattern2 = 55555555
Data Test pattern3 = 33333333
Data Test pattern4 = 0F0F0F0F
Data Test pattern5 = 00FF00FF
Data Test pattern6 = 0000FFFF

4.11.5 TEST 5: CACHE HIT/MISS COMPARITORS TEST -

This test will verify the input paths to the Hit/Miss Comparitors for bits S-A-0, S-A-1, and shorted together, and logic functionality by referencing Index location zero and alternately floating a one across each side of the Hit/Miss Comparitors and checking for misses.

Floating a one across one side of the Hit/Miss Comparitors generates the following test patterns.

pattern 1 = 1000
pattern 2 = 2000
pattern 3 = 4000
pattern 4 = 8000
pattern 5 = 10000
pattern 6 = 20000
pattern 7 = 40000
pattern 8 = 80000
pattern 9 = 100000
pattern 10 = 200000
pattern 11 = 400000
pattern 12 = 800000

Conditional Assembly Block Directives are used in this test to prohibit the test code from referencing Cache Index location zero.

Subtest 1 floats a one across the Cache Tag<23:12> inputs to the Hit/Miss Comparitors. For each pattern generated, zero is compared at the PAD<23:12> inputs to the Comparitors. Patterns 1-6 are tested by performing reads. Patterns 7-12 may be NXM addresses, therefore the IPL is raised to 1F to block out any Write NXM Interrupts, and writes are used to get the pattern into the Cache Tag Store in order to be read out and compared.

Subtest 2 floats a one across the PAD<23:12> inputs to the Hit/Miss Comparitors. Each pattern generated is compared with a Tag Store value of zero read out from Index location zero.

Patterns 1-6 are used by performing read references. Patterns 7 12 are also used by reads but are additionaly set up to expect a Read NXM Machine Check since some of the addresses may be NXM.

4.11.6 TEST 6: CACHE TAG STORE DUAL ADDRESS TEST -

This test checks the Index lines to the Cache Tag Store Rams

for bits S-A-0, S-A-1, and shorted together, by referencing 12 Index locations generated by floating a one across the Index(PAD<11:02>). A Tag Store value of zero is written at Index location zero, and then a Tag Store value of EFF (as many bits set as possible) is written at the next Index location (004). Index location zero is then read again and checked for a hit (a miss could indicate possible shorts in the Index lines). This process continues for writing zeroes each time at Index location zero and then writing EFF into the next Index location. After having written EFF at the other Index locations, zeroes are then written into Index location 004, and EFF will be written into the Tag Store at all other 11 locations. The sequence is controlled by an inner loop pointer and counter for writing EFF into the Index locations, and an outer loop pointer and counter to control writing zeroes into the Index locations.

In order to check all Index lines, two subtests are needed:

the first subtest will use Tag Store values of 000 and EFF.
the second subtest will use Tag Store values of 000 and 100.

NOTE: In this test, a write reference to address EFF000 will always cause an expected Write NXM Interrupt to occur. However, write references to address 100000 may or may not be non-existent. This situation will be handled by setting the NXM_WT_FLAG before writing to address 100000 and not testing the flag after the reference. If the reference is a valid Physical Address, no interrupt will occur, and the test will continue. If, on the other hand, the reference to address 100000 is non-existent, a Write NXM Interrupt will occur. The service routine will first check that the NXM_WT_FLAG is set and then will clear it and return to the test flow, where the flag will not be tested.

The following table of Index values will be used:

CA__INDX__VAL::	000
	004
	008
	010
	020
	040
	080
	100
	200
	400
	800

Conditional Assembly Block Directives are required in this test to prohibit the test code from referencing any of the 11 test Index values.

4.11.7 TEST 7: CACHE DATA STORE DUAL ADDRESS TEST -

This test checks the Index lines to the Cache Data Store Rams for bits S-A-0, S-A-1, and shorted together by floating a one across a field of zeroes in the Index lines of the Data Store and checking for the correct data. A Data Store value of zero is written at Index location zero, then all ones are written into the Data Store at Index location 004. Index location zero is then read and checked for the correct data of all zeroes. This process is then repeated for the 11 Index patterns in the same manner as used in the Cache Tag Store test.

The following table of Index values will be used:

```
CA__INDX__VAL::          000
                         004
                         008
                         010
                         020
                         040
                         080
                         100
                         200
                         400
                         800
```

Conditional Assembly Block Directives are required in this test to prohibit the test code from referencing any of the 11 test Index va'ues.

4.11.8 TEST 8: CACHE BYTE/WORD WRITE MISS TEST -

This test checks that a byte or word write miss will not get cached.

The code first writes a longword of zeroes in Cache at Test location 10000. A byte of ones is then written at test location 11000, which is the same Index position as 10000. However, since this is a byte wrie, the reference should not be written in Cache. A read reference is then made tn location 10000 and it is checked for a hit and correct data. This indicates that the original wrie of zeroes did not get overwritten. The test then repeats for a word write.

Conditional Assembly Block Directives are used in this test to prohibit the test code from referencing Index location zero.

4.11.9 TEST 9: CACHE TAG STORE DYNAMIC MEMORY TEST -

This test verifies the ability to write and read all of the locations in the Cache Tag Store Rams by use of a marching pattern applied across PAD<23:12>. Two Tag Store addresses are used in this test:

Cache Tag Store address 1 = 100

Cache Tag Store address 2 = EFF

The test is divided into four subtests:

Subtest 1 checks the first half of the Cache Tag Store while incrementing through the Index (000-7FC).

Subtest 2 checks the first half of the Cache Tag Store while decrementing through the Index (7FC-000).

Subtest 3 checks the second half of the Cache Tag Store while incrementing through the Index (800-FFC).

Subtest 4 checks the second half of the Cache Tag Store while decrementing through the Index (FFC-800).

Note: At the beginning of the test the IPL is raised to 1F in order to block our Write NXM Interrupts that will occur when referencing Tags of EFF000, and possibly 100000. This eliminates the necessity of making sure that the SCB, SP, and Wt. NXM Service routines do not cause Cache hits that reference Index locations under test.

In addition, Conditional Assembly Block Directives are used to keep the test code in the half of Index locations that is not under test.

4.11.10 TEST 10: CACHE DATA STORE DYNAMIC MEMORY TEST -

This test verifies the ability to write and read all of the locations in the Cache Data Store Rams by use of a marching pattern applied across CACHE<31:00>.

The routine initially writes all the Cache Data Store locations with zeroes (00000000). The first Index location is then read and checked for the correct data, and its complement (FFFFFFFF) is then written into that location. This type of sequence increments and decrements throughout all of the Data Store locations in the Cache.

The starting Physical Address used to reference the Index locations is 10000.

The test is divided into 4 subtests:

Subtest 1 checks the first half of the Data Store locations, (Index 000- 7FC) while incrementing through the Index.

Subtest 2 checks the first half of the Data Store locations, (Index 7FC- 000) while decrementing through the Index.

Subtest 3 checks the second half of the Data Store locations, (Index 800- FFC) while incrementing through the Index.

Subtest 4 checks the second half of the Data Store locations, (Index FFC- 800) while decrementing through the Index.

Conditional Assembly Block Directives are used to keep the test code in the half of Index locations that is not under test.

4.11.11 TEST 11: CACHE BYTE WRITE TEST -

This test will verify that one unique byte of a block of data can be written, while not affecting the other three bytes.

A longword of A's is first written into a test location (10000), and then a byte of ones (FF), is written into byte zero of that test location. The byte write is checked for a hit. A read is then performed on the original test location (10000), and the data is checked for the correct pattern = AAAAAAFF. This method is used for writing to all four bytes of the test location, which verifies the four Enable Byte lines to the Data Store Rams.

The four expected patterns are as follows:

AAAAAAFF
AAAAFFAA
AAFFAAAA
FFAAAAAA

Conditional Assemble Block Directives are used to prohibit the test code from referencing Cache Index location zero.

4.11.12 TEST 12: CACHE I/O SPACE REFERENCE TEST -

This test will verify that a reference to a register in I/O Space will not get cached. The register used to reference I/O

Space will be the Memory Control Register (F20000).

4.11.13 TEST 13: NON CPU MEMORY REFERENCE TEST -

This test will verify that all writes on the CMI which do originate from the CPU will be checked for hits, and those hits will be invalidated. In the case of a miss, no action will be taken.

Five test addresses will be used in this test to check the PAD<23:02> lines.

Cache Test Address 1	=	AAAAA8
Cache Test Address 2	=	555554
Cache Test Address 3	=	CCCCCC
Cache Test Address 4	=	363636
Cache Test Address 5	=	7C03FC

A Unibus Excerciser Terminator(UET) module is required in this test in order to perform writes on the CMI which do not originate from the CPU.

The test is set up as follows:

Bits <23:09> of the Test Address are loaded into the first Unibus Map Register (addr F30800 bits<14:00>), and constitute the Unibus Page Frame Number.

Bits <8:0> of the Test Address are loaded into bits <8:0> of the Unibus Address Register(FFF460) of the UET. Leaving bits <17:09> of the Unibus Address will select the first Unibus Map Register.

The Unibus Control Register of the UET(FFF464) is set up to perform a write. This is accomplished by loading the register with a five(DATO and NPR).

The test then references the first Test Address (AAAAA8) and writes a Tag of AAA into the Tag Store at Index location AA8. The UET then initiates a write to the Test Address. As a result the Index location AA8 should be invalidated. This is confirmed by referencing the same Test Address again, and checking for a Read NXM Machine Check. This Machine Check is the result of the UET reference invalidated the Test Address in the Cache.

This method is repeated for all five patterns.

4.11.14 TEST 14: CACHE LOCKED READ TEST -

This test checks that the read portion of a Locked Read Instruction forces a Cache miss, and gets the data from main memory.

The test code first makes a write reference to Tag Store EFF which will result in a Write NXM Interrupt. The referenced location however, will be cached. A Read Interlocked Instruction (ADAWI) is then performed to that same location. The test is successful is the reference does not go to Cache byt goes to main memory where an expected Read NXM Machine Check will occur. The test fails is the reference goes to Cache where it hits. This hit is verified by checking that the RD_NXM_FLAG is still set.

General TB notes

Most of the TB tests will employ the use of an IPR instruction that will allow Page Table Entries (PTE s) to be written into and read out of the TB. This allows almost all of the TB hardware to be tested without having to enable Memory Management.

In order to write a PTE into the TB, the desired Virtual Address (VA) must be loaded into the P0 Base Register (P0BR). The P0BR is used only to load the VA and does not function as a mapping register in this application. Once the VA has been loaded, the PTE is loaded as follows:

```
MTPR    _#PTE,_#TB__DATA        ; Writes desired PTE into the TB
                                ; at VA spec in P0BR.
or

MFPR    _#TB__DATA,RX           ; Reads PTE at VA spec. by P0BR
                                ; into desired register.
```

A TB hit is verified by reading the PTE and checking that the Valid Bit, PTE<31> is set. The Valid bit is the XOR of the hit/miss comparitors of both TB groups.

The PTE is comprised from the following:

Bit_<31_>	Valid Bit	from PAD_<08_>
Bits_<30:27_>	Protection code	from PAD_<07:04_>
Bit <26_>	Modified bit	from PAD_<03_>
Bit_<25:15_>	Unused bits, these bits are indeterminate and are masked out when reading a PTE for data checking.	
Bits <14:00_>	Page Table Entry (PTE)	from PAD_<23:09_>

The TB Tag Store Address is comprised of Memory Address(MAD)<30:16>.

The TB Tag Store is indexed by MAD<31,15:09>.

Extensive use is made of the Force Replace and Disable bits in the TB Group Disable Register (TBDR)

Translation Buffer Invalidate Single (TBIS) will be used where possible within subtests for purposes of shortening scope loops. TB Flushes (Invalidate all), will still be used outside of the subtests.

SPECIAL TB TROUBLE-SHOOTING TEST

A general purpose subtest exists for the TB to write, read, and loop on any desired VA and PTE. To use this routine perform the following:

Deposit desired VA into R5
Deposit desired PTE into R1
Deposit desired replacement code into R6 as follows:

hex A = replace Group 0, disable Group 1
hex D = replace Group 1, disable Group 0
0 = enable Random Replacement

Start routine a address 220; the test will continually loop unless the BRB instruction is replaced with a NOP.

Six additional NOP s are also provided to modify the routine.

4.11.15 TEST 15: TB CONTROL LOGIC TEST -

Subtests 1-3 check TB maintenance register bits for read/write capability. Subtests 4-9 attempts simple hits and misses in TB Group 0, and subtests 10-15 attempts same in TB Group 1. Subtests 16,17 check to multiple hits.

Subtest 4 checks that a TB hit in Group 0 can be verified by writing and reading a PTE with only the Valid Bit Set.

Subtest 5 checks that a TB miss in Group 0 can be verified by writing and reading a PTE with the Valid bit clear.

Subtest 6 checks that all bits of the TB Tag Store in Group 0 can be set by checking for a TB hit with a VA = 7FFF0000.

Subtest 7 checks that all bits of the VA Tag Store in Group 0 can be cleared by checking for a TB hit at VA = 0.

Subtest 8 checks that all bits of the TB Data Store in Group 0 can be set by writing and reading a PTE = FC007FFF. The indeterminate bits, PTE<25 :15> are shown as zeroes for simplicity.

Subtest 9 checks that all bits of the TB Data Store in Group 0 can be cleared by writing and reading a PTE = 80000000. Note that PTE<31> must be set (Valid) to check the whole PTE.

Subtests 10-15 are the same as 4-9 except that Group 1 is used.

Subtest 16 checks for occurence of TB multiple hits while forcing replacement to Group 0 and not disabling either group. A PTE is then written at VA=0 and checked for a hit. A miss indicates that a hit was not detected or multiple hits occured.

Subtest 17 checks for occurence of TB multiple hits while forcing replacement to Group 1 and not disabling either group. A PTE is then written at VA=0 and checked for a hit. A miss indicates that a hit was not detected or multiple hits occured.

4.11.16 TEST 16: TB ADDRESS PATH TAG MEMORY TEST GROUP 0

This test checks the input and output paths of Group 0 s TB Tag Store for bits S-A-0, S-A-1, and shorted together. This is accomplished by forcing replacement of TB Group 0 and referencing TB Index location zero VA<31,15:09>=0, while writing a minimal bit fault pattern set to the TB Tag Store, VA<30:16> and checking for hits. The parity generator, checker, and hit/miss comparitors of Group 0 are implicitely checked in this test.

Five test patterns are used in this test, they are as follows:

TB pattern 1	55550000
TB pattern 2	2AAA0000
TB pattern 3	33330000
TB pattern 4	0F0F0000
TB pattern 5	00FF0000

4.11.17 TEST 17: TB ADDRESS PATH TAG MEMORY TEST GROUP 1

This test checks the input and output paths of Group 1's TB Tag Store for bits S-A-0, S-A-1, and shorted together. This is accomplished by forcing replacement of TB Group 1 and referencing TB Index location zero VA<31,15:09>=0, while writing a minimal bit fault pattern set to the TB Tag Store, VA<30:16> and checking for hits. The parity generator, checker, and hit/miss comparitors of Group 1 are implicitely checked in this test.

Five test pattern are used in this test, they are as follows:

TB pattern 1	55550000
TB pattern 2	2AAA0000
TB pattern 3	33330000
TB pattern 4	0F0F0000
TB pattern 5	00FF0000

4.11.18 TEST 18: TB DATA PATH DATA MEMORY TEST GROUP 0

This test checks the input and output paths of Group 0's TB Data Store for bits S-A-0, S-A-1, and shorted together. This is accomplished by forcing replacement of TB Group 0 and referencing TB Index location zero (VA=0), and writing a minimal bit fault pattern set into the PTE, and checking for the correct data results. This test also checks the TB Data Store Output Mux.

Five patterns are required for this test, they are as follows:

TB pattern 1	D4005555
TB pattern 2	A8002AAA
TB pattern 3	BC000F0F
TB pattern 4	FC0000FF

4.11.19 TEST 19: TB DATA PATH DATA MEMORY TEST GROUP 1

This test checks the input and output paths of Group 1 s TB Data Store for bits S-A-0, S-A-1, and shorted together. This is accomplished by forcing replacement of TB Group 1 and referencing TB Index location zero (VA=0), and writing a minimal bit fault pattern set into the PTE, and checking for the correct data results. This test a'so checks the TB Data Store Output Mux.

Five patterns are required for this test, they are as follows:

TB pattern 1 D4005555
TB pattern 2 A8002AAA
TB pattern 3 BC000F0F
TB pattern 4 FC0000FF

4.11.20 TEST 20: TB HIT/MISS COMPARITORS TEST GROUP 0 -

This test will verify the inputs to the Hit/Miss Comparitors (='s checker) of TB Group 0 by referencing TB Index location zero (VA=0), and alternately floating a one across a field of zeroes on each input side of the comparitors while checking for misses.

4.11.21 TEST 21: TB HIT/MISS COMPARITORS TEST GROUP 1 -

This test will verify the inputs to the Hit/Miss Comparitors (= s checker) of TB Group 1 by referencing TB Index location zero (VA=0), and alternately floating a one across a field of zeroes on each input side of the comparitors while checking for misses.

4.11.22 TEST 22: TB DUAL ADDRESS TAG STORE TEST GROUP 0 -

This test checks the TB Index lines, VA<31,15:09>, to the Tag Store of Group 0 for bits S-A-0, S-A-1, and shorted together by referencing Index locations generated by floating a one across a field of zeroes in the Index.

Nine TB Index locations are generated in this test, they are as follows:

TB Index location 1 = 0
TB Index location 2 = 200
TB Index location 3 = 400
TB Index location 4 = 800
TB Index location 5 = 1000
TB Index location 6 = 2000
TB Index location 7 = 4000
TB Index location 8 = 8000
TB Index location 9 = 80000000

The test sequence is as follows:

Zeroes are written into the TB Tag Store at Index location 000. Location zero is then read and checked for a hit. If successful, ones are written into the next Index location (400), and location zero is read again and checked for a hit. This process continues for writing ones into the remaining Index locations, each time rereading location zero and checking for a hit. The sequence is controlled by an Inner Loop pointer and counter. When the Inner Loop is exited, zeroes are then written into Index location 200, and ones are written to all the other Index locations. An Outer Loop pointer and counter controls the sequence of writing zeroes, and then control passes to the Inner Loop for the writing of ones.

4.11.23 TEST 23: TB DUAL ADDRESS TAG STORE TEST GROUP 1 -

This test checks the TB Index lines, VA<31,15:09>, to the Tag Store of Group 1 for bits S-A-0, S-A-1, and shorted together by referencing Index locations generated by floating a one across a field of zeroes in the Index.

Nine TB Index locations are generated in this test, they are as follows:

TB Index location 1 = 0
TB Index location 2 = 200
TB Index location 3 = 400
TB Index location 4 = 800
TB Index location 5 = 1000
TB Index location 6 = 2000
TB Index location 7 = 4000
TB Index location 8 = 8000
TB Index location 9 = 80000000

The test sequence is as follows:

Zeroes are written into the TB Tag Store at Index location 000. Location zero is then read and checked for a hit. If successful, ones are written into the next Index location (400), and location zero is read again and checked for a hit. This process continues for writing ones into the remaining Index locations, each time rereading location zero and checking for a hit. The sequence is controlled by an Inner Loop pointer and counter. When the Inner Loop is exited, zeroes are then written into Index location 200, and ones are written to all the other Index locations. An Outer Loop pointer and counter controls the sequence of writing zeroes, and then control passes to the Inner Loop for the writing of ones.

4.11.24 TEST 24: TB DUAL ADDRESS DATA STORE TEST GROUP 0 -

This test checks the TB Index lines to the Data Store Rams of Group 0 for bits S-A-0, and S-A-1, and shorted together. Zeroes are written into the PTE in the TB at Index location zero, then ones are written into the PTE at Index location 200. Index location zero is then read, and the PTE is checked for the correct data. This process is continued in the same manner as the TB Dual Address Tag Store Tests (tests 22 and 23).

Nine TB Index locations are generated in this test, they are as follows:

TB Index location 1 = 0
TB Index location 2 = 200
TB Index location 3 = 400
TB Index location 4 = 800
TB Index location 5 = 1000
TB Index location 6 = 2000
TB Index location 7 = 4000
TB Index location 8 = 8000
TB Index location 9 = 80000000

4.11.25 TEST 25: TB DUAL ADDRESS DATA STORE TEST GROUP 1 -

This test checks the TB Index lines to the Data Store Rams of Group 1 for bits S-A-0, and S-A-1, and shorted together. Zeroes are written into the PTE in the TB at Index location zero, then ones are written into the PTE at Index location 200. Index location zero is then read, and the PTE is checked for the correct data. This process is continued in the same manner as the TB Dual Address Tag Store Tests (tests 22 and 23).

Nine TB Index locations are generated in this test, they are as follows:

TB Index location 1 = 0
TB Index location 2 = 200
TB Index location 3 = 400
TB Index location 4 = 800
TB Index location 5 = 1000
TB Index location 6 = 2000
TB Index location 7 = 4000
TB Index location 8 = 8000
TB Index location 9 = 80000000

4.11.26 TEST 26: TB DYNAMIC TEST TAG STORE MEMORY GROUP 0

This test verifies the ability to write and read all of the 256 decimal locations in the Tag Store Rams of TB Group 0 by use of a marching pattern applied to the VA Tag Store.

The test consists of two subtests.

Subtest 1 accomplishes the following:
Increments through the 256 Tag Store locations and fills them with a VA Tag = zero.
Increments through the 256 Tag Store locations each time reading and checking for a VA=zero and then writing the complement (VA=all ones) into that same location.
Increments through the 256 Tag Store locations each time reading and checking for a VA=all ones and then writing the complement (VA=zeroes) into that same locaton.

Subtest 2 accomplishes the following:
Decrements through the 256 Tag Store locations each time reading and checking for a VA=zeroes and then writing the complement (VA=all ones) into that same location.
Decrements through the 256 Tag Store locations each time reading and checking for a VA=all ones and then writing the complement (VA=zeroes) into that same location.

4.11.27 Test 27: TB DYNAMIC TEST TAG STORE MEMORY GROUP 1

This test verifies the ability to write and read all of the 256 decimal locations in the Tag Store Rams of TB Group 1 by use of a marching pattern applied to the VA Tag Store.

The test consists of two subtests.

Subtest 1 accomplishes the following:
Increments through the 256 Tag Store locations and fills them with a VA Tag = zero.
Increments through the 256 Tag Store locations each time reading and checking for a VA=zero and then writing the complement (VA=all ones) into that same location.
Increments through the 256 Tag Store locations each time reading and checking for a VA=all ones and then writing the complement (VA=zeroes) into that same locaton.

Subtest 2 accomplishes the following:
Decrements through the 256 Tag Store locations each time reading and checking for a VA=zeroes and then writing the complement (VA=all ones) into that same location.
Decrements through the 256 Tag Store locations each time

reading and checking for a VA=all ones and then writing the complement (VA=zeroes) into that same location.

4.11.28 TEST 28: TB DYNAMIC TEST DATA STORE MEMORY GROUP 0

This test verifies the ability to write and read all of the locations in the TB Data Store Rams of Group 0 by use of a marching pattern applied to the PTE.

The test consists of two subtests.

Subtest 1 accomplishes the following:
Increments through the 256 Data Store locations filling them with zeroes.
Increments through the 256 Data Store locations reading and checking for a PTE=8000000 and then writing a PTE=FC007FFF into that same location.
Increments through the 256 Data Store locations reading and checking for a PTE=FC007FFF and then writing a PTE=80000000 into that same location.

Subtest 2 accomplishes the following:
Decrements through the 256 Data Store locations reading and checking for a PTE=80000000 and then writing a PTE=FC007FFF into that same location.
Decrements through the 256 Data Store locations reading and checking for a PTE=FC007FFF and then writing a PTE=80000000 into that same location.

4.11.29 TEST 29: TB DYNAMIC TEST DATA STORE MEMORY GROUP 1

This test verifies the ability to write and read all of the locations in the TB Data Store Rams of Group 1 by use of a marching pattern applied to the PTE.

The test consists of two subtests.

Subtest 1 accomplishes the following:
Increments through the 256 Data Store locations filling them with zeroes.
Increments through the 256 Data Store locations reading and checking for a PTE=8000000 and then writing a PTE=FC007FFF into that same location.
Increments through the 256 Data Store locations reading and checking for a PTE=FC007FFF and then writing a PTE-80000000

into that same location.

Subtest 2 accomplishes the following:
Decrements through the 256 Data Store locations reading and checking for a PTE=80000000 and then writing a PTE=FC007FFF into that same location.
Decrements through the 256 Data Store locations reading and checking for a PTE=FC007FFF and then writing a PTE=80000000 into that same location.

4.11.30 TEST 30: TB DYNAMIC TEST OF VALID BIT GROUP 0 -

This test verifies the ability to write and read all of the Valid Bit locations in TB Tag Store of Group 0 by use of a marching pattern applied to the Valid Bit.
The test consists of two subtests.

Subtest 1 accomplishes the following:
Increments through the 256 Valid Bit locations each time reading and checking for a clear Valid Bit and then setting that bit.
The Valid Bits are initially cleared by virtue of the TB flush.
Increments through the 256 Valid Bit locations each time reading and checking for a set Valid Bit and then clearing that bit.

Subtest 2 accomplishes the following:
Decrements through the 256 Valid Bit locations each time reading and checking for a clear Valid Bit and then setting that bit.
Decrements through the 256 valid Bit locations each time reading and checking for a set Valid Bit and then clearing that bit.

4.11.31 TEST 31: TB DYNAMIC TEST OF VALID BIT GROUP 1 -

This test verifies the ability to write and read all of the Valid Bit locations in TB Tag Store of Group 1 by use of a marching pattern applied to the Valid Bit.
The test consists of two subtests.

Subtest 1 accomplishes the following:
Increments through the 256 Valid Bit locations each time reading and checking for a clear Valid Bit and then setting that bit.
The Valid Bits are initially cleared by virtue of the TB

flush.
Increments through the 256 Valid Bit locations each time reading and checking for a set Valid Bit and then clearing that bit.

Subtest 2 accomplishes the following:
Decrements through the 256 Valid Bit locations each time reading and checking for a clear Valid Bit and then setting that bit.
Decrements through the 256 valid Bit locations each time reading and checking for a set Valid Bit and then clearing that bit.

4.11.32 TEST 32: TB RANDOM GROUP REPLACEMENT TEST -

This test checks that random replacement occurs in both TB Groups. A TB Flush is first performed to invalidate all TB locations. Random Replacement is then enabled and 256 TB locations are written with a PTE=80000000. Next, Group 1 is disabled and the same 256 locations are read, looking for a hit which would indicate random replacement to Group 0. If successful, Group 0 is disabled and those same 256 locations are read, this time looking for a hit in Group 1.

The subroutine INC_INDEX is used in this test to generate the TB Index locations.

NOTE: The Cache is disabled for this test only, in order to force all references to go to main memory where refresh stalls will occur. This will cause the TB Group Replacement Flip Flop to toggle more randomly.

MAPPING INITIALIZATION ROUTINE

This routine sets up the Virtual and Physical Address Space necessary for a Mapping test to be performed. Physical Address Space starting at 64-Kilobytes will be allocated for the System Page Table(SPT), Program Page Table(P0PT), and Control Page Table(P1PT). All PTE's will be set for Read/Write access. Each PTE will contain Page Frame Numbers (PFN s) from 0 to FF. The following Base and Length Registers will be loaded with information that will allow a System Space Virtual Address, Program Space Virtual Address, and a Control Space Virtual Address to map to the same Physical Address in Main Memory.

System Base Register (SBR) = 00010000
System Length Register (SLR) = 100
P0 Base Register (P0BR) - 80011000

```
P0 Lenght Register        (P0LR)  =          100
P1 Base Register  (P1BR)  =       80012000
P1 Lenght Register        (P1LR)  =            0
```

4.11.33 TEST 33: TB MAPPING TEST -

This test verifies proper TB operation with Memory Mapping enabled. Three Virtual Addresses designated as VA1, VA2, and VA3 will be referenced causing all three to map to Physical Address 18000.

The three Virtual Addresses used are as follows:

```
VA Program Space        00018000
VA System Space         80018000
VA Control Space        40018000
```

Three test patterns are used starting at the label "TB__PAT__D":

```
Test Pattern 1            AAAAAAAA
Test Pattern 2            55555555
Test Pattern 3            F0F0F0F0
```

The test consists of four subtests:

Subtest 1 checks that the correct Physical Address will be referenced each time in Main Memory when reading VA1,VA2, or VA3. This is accomplished by writing all ones in PA 018000, enabling Mapping, and reading and checking VA1-3 for correct data results.

Subtest 2 checks that the same test pattern will be read when writing to either of the three VA's. This is accomplished by writing Test Pattern 1 into VA1 and then reading all three VA's and checking for the correct data results. Test Pattern 2 is then written into VA2 and all three VA's are read and checked. Finally Test Pattern 3 is written into VA3 and all three VA s are read and checked for correct data.

Subtest 3 is identical to subtest 2 except that Force Replacement is done on TB Group 1, and Group 0 is disabled.

Subtest 4 is identical to subtest 2 except the Random Replacement is used.

4.11.34 TEST 34: TB MODIFIED BIT TEST -

This test verifies that the Modified bit will set on a valid, access allowed write to memory.

This is accomplished by setting up a System Page Table Entry (SPTE) with the M Bit<26> clear (SPTE=A00000C0). The page is then written into and the M bit is checked.

4.11.35 TEST 35: TB CROSS PAGE REFERENCE TEST -

This test makes memory references across two pages designated as VPN1 (VA=00018000) and VPN2 (VA=00018200). Longword references to the last two bytes in VPN1 will cause the first two bytes in VPN2 to be referenced. Using this technique, each page will be set up with various access rights and the appropriate response will be checked.

The PTE for VPN1 is at PA=011300.
The PTE for VPN2 is at PA=011304.

Prior to making any cross page reference, an Access Control Violation Flag (known as ACC_VIO_FLAG) will be set. The reference is then attempted. If an Access Control Violation (ACV) is expected, the service routine will clear the ACC_VIO_FLAG and return from the exception. The test routine will check the flag, if clear, it indicates that the expected Access Control Violation did in fact, occur. If the flag was still set, the exception never occured and an error halt would result.

Inputs used in this test are as follows:

SPTE	=	010300	Physical location of the PTE in the SPT
SYS_PTBL	=	010000	Physical location of SPT
P0_PTBL	=	80011000	Virtual location of P0PT
P1_PTBL	=	80012000	Virtual location of P1PT
P0PTE_VPN1	=	011300	Physical location of PTE
P0PTE_VPN2	=	181FE	Byte 511 of VPN1

The test is comprised of five subtests:

Subtest 1 does a write and read of a longword of A's across two pages set up with read/write access. No Access Control Violation should occur.

Subtest 2 attempts to write a longword of 5 s across two pages. VP1 is set up for read/write access, and VP2 is read only. An Access Control Violation should occur and the write should not be allowed. A read is then attempted across both

pages which should be allowed, and the data results should still indicate the same pattern of A s which were written from the previous subtest.

Subtest 3 attempts to write a longword of 5's across two pages. VP1 is set up for read only access, and VP2 is read/write access. An Access Control Violation shuld occur and the data should not be written. A read is then attempted to verify th previous data pattern of A s.

Subtest 4 attempts to read across two pages. VP1 is set up with read only access, and VP2 is set up for no access. An Access Control Violation should occur.

Subtest 5 is set to to read across two pages. VP1 is set up with no access, and VP2 is set up for read only access. An Access Control Violation should occur.

5 MAINTENANCE HISTORY

Version 3.4 Susan E. Hutcheson Novemeber 14, 1984 Corrected version number displayed in endpass message.

Object Module Synopsis

Module Name	Ident	Bytes	File	Creation Date	Creator
ECKAL	3 4	32933	DRB2: HUTCHESON.ECKAL ECKAL.OBJ;1	13-NOV-1984 15:07	VAX/VMS Macro V04 00

Program Section Synopsis

Psect Name	Module Name	Base	End	Length	Align	Attributes
CACHTB		00000000	000080A4	000080A5 (32933.)	PAGE 9	NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC
	ECKAL	00000000	000080A4	000080A5 (32933.)	PAGE 9	

Symbols By Name

Symbol	Value	Symbol	Value	Symbol	Value
ACC_VIO_FLAG	00000A06-R	NXM_RD_FLAG	00000A05-R	SCB20	00000420 R
APT_CMD_WORD	0000FE04	NXM_WT_FLAG	00000A04-R	SCB24	00000424 R
APT_COMMAND	00000C82-R	P0BR	0000000B	SCB60	00000460-R
APT_CTRL_FLAGS	0000FE00	P0LR	00000009	SCB64	00000464-R
APT_ERR_FLAG	0000FE40	P0PTE_VPN1	00011300	SCB8	00000408-R
APT_ERR_PC	0000FE44	P0PTE_VPN2	00011304	SCBB	00000011
APT_MSG_TYPE	0000FE40	P0_PTBL	80011000	SECONDARY_REV	00000003
APT_PASS_CNT	0000FE54	P1BR	0000000A	SLR	0000000D
APT_TEST_COUNT	0000FE52	P1LR	0000000B	SPEC_TB_TEST	00000A3B-R
APT_TEST_NUM	0000FE50	P1_PTBL	80012000	SPTE	00010300
BOT_OF_STACK	00000800-R	PCS_BITLOAD	000053C0-R	SYNC_POINT	00FFF460
BUS_ER_REG	00000024	PCS_INIT	00005384-R	SYS_PTBL	00010000
CADR	00000025	PCS_INIT_SAVE	00000298-R	TBDR	00000024
CAER	00000027	PCS_MCHK_HANDL	00005448-R	TBIA	00000039
CA_ER_BITS	0000000E	PCS_MICLOAD	000053F1-R	TBIS	0000003A
CA_ER_REG	00000020	PCS_PATCH_END	000080A5-R	TB_DATA	0000003B
CA_INDX_VAL	0000029C-R	PCS_PRESENT	0000539A-R	TB_ER_BIT	00000002
CA_TST_ADR	000002C8-R	PCS_PTH_BIT_ARRAY	000054A5-R	TB_PAT_A	000002F0-R
CONT_PC	00000A0B-R	PCS_PTH_MIC_ARRAY	000058A5-R	TB_PAT_B	00000304-R
DEC_INDEX	00000A25-R	PCS_SAVE_COUNT	00000294-R	TB_PAT_C	00000318-R
ENDPASS	0000024C-R	PCS_STATUS_FLAGS	00000223-R	TB_PAT_D	0000033C-R
END_PASS	00000C4C-R	PRIMARY_REV	00000003	TB_VAL	0000001F
END_VPN1	000181FE	PTE_MASK	03FF8000	TCB	00000600 R
ERROR_CALL	00005484-R	RD_NXM	00000003	TCB0	00000600-R
INC_INDEX	00000A0F-R	READY	00000007	TCB10	00000610 R
IO_RESET	00000037	RET_PC	00000A07-R	TCB100	00000700-R
IPL	00000012	RXCS	00000020	TCB14	00000614 R
ISP	00000004	RXDB	00000021	TCB18	00000618 R
KSP	00000000	SBR	0000000C	TCB1C	0000061C R
MA_CHK_REG	00000028	SCB	00000400-R	TCB20	00000620 R
MCESR	00000026	SCB0	00000400-R	TCB24	00000624 R
MME	00000038	SCB04	00000404-R	TCB28	00000628 R

Symbol	Value	Symbol	Value	Symbol	Value
TCB2C	0000062C-R	TEST_28	000045A4-R		
TCB30	00000630-R	TEST_29	00004791-R		
TCB34	00000634-R	TEST_3	000012E0-R		
TCB38	00000638-R	TEST_30	0000497E-R		
TCB3C	0000063C-R	TEST_31	00004AEA-R		
TCB4	00000604-R	TEST_32	00004C57-R		
TCB40	00000640-R	TEST_33	00004DC1-R		
TCB44	00000644-R	TEST_34	00004F9D-R		
TCB48	00000648-R	TEST_35	00005026-R		
TCB4C	0000064C-R	TEST_4	0000150A-R		
TCB50	00000650-R	TEST_5	0000172E-R		
TCB8	00000608-R	TEST_6	000019F1-R		
TCB84	00000684-R	TEST_7	00001B5F-R		
TCB88	00000688-R	TEST_8	00001C19-R		
TCB8C	0000068C-R	TEST_9	00001D31-R		
TCB90	00000690-R	TOP_OF_STACK	00000A00-R		
TCB94	00000694-R	TXCS	00000022		
TCB98	00000698-R	TXDB	00000023		
TCB9C	0000069C-R	UET_ADDR	00FFF460		
TCBA0	000006A0-R	UET_CNTL	00FFF464		
TCBA4	000006A4-R	UNI_MAP	00F30800		
TCBA8	000006A8-R	UNI_PFN	000002DC-R		
TCBAC	000006AC-R	VA1	00006348-R		
TCBB0	000006B0-R	VA2	0000034C-R		
TCBB4	000006B4-R	VA3	00000350-R		
TCBB8	000006B8-R				
TCBBC	000006BC-R				
TCBC	0000060C-R				
TCBC0	000006C0-R				
TCBC4	000006C4-R				
TCBF8	000006F8-R				
TCBFC	000006FC-R				
TEST_1	00000D70-R				
TEST_10	00002168-R				
TEST_11	00003150-R				
TEST_12	0000320E-R				
TEST_13	000032E8-R				
TEST_14	0000349D-R				
TEST_15	0000353D-R				
TEST_16	00003A0E-R				
TEST_17	00003A99-R				
TEST_18	00003B27-R				
TEST_19	00003BB7-R				
TEST_2	000011F4-R				
TEST_20	00003C47-R				
TEST_21	00003D8A-R				
TEST_22	00003ED1-R				
TEST_23	00003F94-R				
TEST_24	00004057-R				
TEST_25	00004118 R				
TEST_26	000041DC-R				
TEST_27	000043C0 R				

Symbol	Value	Symbol	Value	Symbol	Value

Key for special characters above:

*	- Undefined
U	- Universal
R	- Relocatable
X	- External

Image Synopsis

```
Virtual memory allocated:                  00000000 000081FF 00008200 (33280. bytes, 65. pages)
Stack size:                                       0. pages
Image binary virtual block limits:                1.       65. (   65. blocks)
Image name and identification:             ECKAL 3-4
Number of files:                                  1.
Number of modules:                                1.
Number of program sections:                       2.
Number of global symbols:                       173.
Number of image sections:                         1.
Image type:                                SYSTEM.
Map format:                                DEFAULT in file DRB2: HUTCHESON.ECKAL ECKAL.MAP;1
Estimated map length:                      19. blocks
```

Link Run Statistics

Performance Indicators	Page Faults	CPU Time	Elapsed Time
Command processing:	90	00:00:00.16	00:00:00.18
Pass 1:	41	00:00:00.31	00:00:00.64
Allocation/Relocation:	9	00:00:00.09	00:00:00.29
Pass 2:	57	00:00:00.93	00:00:01.70
Map data after object module synopsis:	10	00:00:00.39	00:00:00.89
Symbol table output:	0	00:00:00.02	00:00:00.16
Total run values:	207	00:00:01.90	00:00:03.86

```
Using a working set limited to 600 pages and 44 pages of data storage (excluding image)

Tota. number object records read (both passes):   158
    of which 0 were in libraries and 2 were DEBUG data records containing 38 bytes

Number of modules extracted explicitly              = 0
    with 0 extracted to resolve undefined symbols

0 library searches were for symbols not in the library searched

A total of 0 global symbol table records was written

LINK/SYST-0 ECKAL
```

Table of contents

(1) 71 DECLARATIONS
(1) 83 SEQ
(1) 412 Messages used by this program
(1) 424 Own Storage
(1) 586 Special TB Trouble-Shooting test
(1) 630 INITIALIZATION ROUTINE
(1) 689 Machine Check Service Routine
(1) 763 Write Non_Existent Mem Routine
(1) 836 Access Control Violation Rtn
(1) 875 End Pass Routine
(1) 903 APT Command Routine
(1) 929 Check APT Command Routine
(1) 962 Type End Pass Message Routine
(1) 974 Print Character Routine
(1) 984 Print DS> Prompt to APT Routine
(1) 997 Type 'No PCS' to user Routine
(1) 1010 General Notes
(1) 1036 TEST 01: CACHE CONTROL LOGIC TEST
(2) 1241 TEST 02: NON-EXISTENT MEMORY REFERENCE TEST
(2) 1308 TEST 03: SIMPLE ADDRESS PATHS TEST
(2) 1501 TEST 04: SIMPLE DATA PATHS TEST
(2) 1673 TEST 05: HIT/MISS COMPARITORS TEST
(3) 1940 TEST 06: TAG STORE DUAL ADDRESS TEST
(4) 2123 TEST 07: DATA STORE DUAL ADDRESS TEST
(5) 2235 TEST 08: BYTE/WORD WRITE MISS TEST
(6) 2339 TEST 09: CACHE TAG STORE DYNAMIC MEMORY TEST
(7) 2687 TEST 10: CACHE DATA STORE DYNAMIC MEMORY TEST
(8) 3035 TEST 11: BYTE WRITE TEST
(9) 3116 TEST 12: CACHE I/O SPACE REFERENCE TEST
(10) 3184 TEST 13: NON CPU MEMORY REFERENCE TEST
(11) 3298 TEST 14: LOCKED READ TEST
(11) 3356 TEST 15: TB CONTROL LOGIC TEST
(11) 3780 TEST 16: ADDRESS PATH TAG MEMORY TEST GROUP 0
(11) 3830 TEST 17: ADDRESS PATH TAG MEMORY TEST GROUP 1
(11) 3881 TEST 18: TB DATA PATH DATA MEMORY TEST GROUP 0
(11) 3932 TEST 19: TB DATA PATH DATA MEMORY TEST GROUP 1
(11) 3985 TEST 20: TB HIT/MISS COMPARITORS TEST GROUP 0
(11) 4083 TEST 21: TB HIT/MISS COMPARITORS TEST GROUP 1
(11) 4182 TEST 22: TB DUAL ADDRESS TAG STORE TEST GROUP 0
(11) 4261 TEST 23: TB DUAL ADDRESS TAG STORE TEST GROUP 1
(11) 4344 TEST 24: TB DUAL ADDRESS DATA STORE TEST GROUP 0
(11) 4420 TEST 25: TB DUAL ADDRESS DATA STORE TEST GROUP 1
(11) 4495 TEST 26: TB DYNAMIC TEST TAG STORE MEMORY GROUP 0
(11) 4658 TEST 27: TB DYNAMIC TEST TAG STORE MEMORY GROUP 1
(11) 4819 TEST 28: TB DYNAMIC TEST DATA STORE MEMORY GROUP 0
(11) 4988 TEST 29: TB DYNAMIC TEST DATA STORE MEMORY GROUP 1
(11) 5159 TEST 30: TB DYNAMIC TEST OF VALID BIT GROUP 0
(11) 5292 TEST 31: TB DYNAMIC TEST OF VALID BIT GROUP 1
(11) 5427 TEST 32: TB RANDOM GROUP REPLACEMENT TEST
(11) 5509 MAPPING INITIALIZATION ROUTINE
(11) 5555 TEST 33: TB MAPPING TEST
(11) 5736 TEST 34: TB MODIFIED BIT TEST
(11) 5768 TEST 35: TB CROSS PAGE REFERENCE TEST
(11) 6028 INITIALIZATION ROUTINE
(11) 6072 PCS_PRESENT - Check for the presense of a PCS (L0008) module
(11) 6115 PCS_BITLOAD - Load the patch bits
(11) 6156 PCS_MICLOAD - Load the microcode patches

Table of contents

(11) 6209 PCS_MCHK_HANDL - Machine handler routine.
(11) 6270 Patch bit and micro word area

```
                        1  .TITLE ECKAL VAX 11/750 CACHE/TB DIAGNOSTIC PROGRAM
                        2  .IDENT /3-4/
                        3
                        4 ;
                        5 ; COPYRIGHT (C) 1980, 1983
                        6 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
                        7 ;
                        8 ; THIS SOFTWARE IS FURNISHED  UNDER A LICENSE FOR USE ONLY ON A SINGLE
                        9 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
                       10 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
                       11 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
                       12 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
                       13 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
                       14 ; REMAIN IN DEC.
                       15 ;
                       16 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
                       17 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
                       18 ; CORPORATION.
                       19 ;
                       20 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
                       21 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                       22 ;
                       23
                       24 ;++
                       25 ; FACILITY:
                       26 ;
                       27 ; Vax Diagnostic Library
                       28 ;
                       29 ;
                       30 ; ABSTRACT:
                       31 ;
                       32 ; This program provides quick verification of the Cache/TB
                       33 ; portion of a VAX 11/750 CPU. Hardware errors are isolated
                       34 ; to the function, with a halt occuring on error. The program
                       35 ; is macro level four, which operates stand alone without
                       36 ; the use of the Diagnostic Supervisor.
                       37 ;
                       38 ;
                       39 ;
                       40 ; ENVIRONMENT:
                       41 ;
                       42 ; Stand-alone
                       43 ;
                       44 ;
                       45 ;
                       46 ;
                       47 ; AUTHOR                    , CREATION DATE:
                       48 ;
                       49 ; Domenic Andella             18-Oct-79
                       50 ;
                       51 ;
                       52 ;
                       53 ;
                       54 ; MODIFIED BY: Domenic Andella 11-Jun-80
                       55 ;
                       56 ; 2.0 Installed Cache/TB enhancements as indicated in
                       57 ;  memo on 14-Nov-79, as a result of diagnostic
```

```
0000     58 ;  code review.
0000     59 ; 3.0 Don Monroe February 1981
0000     60 ;  Updated to support version 62 micro code.
0000     61 ; 3.1 Dan Milleville January, 1982
0000     62 ;  Updated tests 3 & 13 to accomodate up to 8M of memory.
0000     63 ;
0000     64 ; 3.2B Dan Milleville Jun , 1983
0000     65 ;  New version of 3.2 to clear up the problem of a glitched
0000     66 ;  version.
0000     67 ;
0000     68 ; 3.4     Susan E. Hutcheson
0000     69 ;  Corrected version number for endpass message.
0000     70 ;
```

```
                        71   .SBTTL  DECLARATIONS
00000000                72   .PSECT  CACHTB,PAGE
                        73
                        74
                        75 ;
                        76 ;LISTING CONTROL
                        77 ;
                        78   .LIST ME
                        79 ;
                        80 ; MACROS:
                        81 ;
                        82
                        83   .SBTTL  SEQ
                        84 ;++
                        85 ; FUNCTIONAL DESCRIPTION:
                        86 ;
                        87 ;  MACRO SEQ
                        88 ;
                        89 ;    THIS MACRO GENERATES THE FOLLOWING:
                        90 ;
                        91 ;    PSECT FOR EACH TEST,
                        92 ;    JUMP INSTR. TO SEQUENCE TO NEXT TEST,
                        93 ;    FIRST TEST NUMBER LABEL FOR EACH TEST,
                        94 ;    FIRST SUBTEST LABEL,
                        95 ;    EXECUTION SEQUENCER TEST
                        96 ;
                        97 ;
                        98 ;--
                        99
                       100
                       101   .MACRO SEQ TST_NO  ; Macro Call for Sequencer
                       102   .NLIST    ;
                       103   $TN=$TN+1   ; Increment the Test Number
                       104   .LIST    ;
                       105 TEST_'TST_NO'::    ; Continued on next line
                       106   MOVW #'TST_NO',APT_TEST_NUM ; Load the Test Number
                       107   INCW APT_TEST_COUNT  ; Bump up the test count
                       108   CMPW APT_TEST_NUM,-  ; Continued on next line
                       109    APT_TEST_COUNT  ; Check for the correct test number
                       110   BEQL T'TST_NO'_S1   ; Sequence correct, continue test
                       111   BBC #31,APT_CTRL_FLAGS,-
                       112    T'TST_NO_HLT   ; Branch if not APT mode
                       113   MOVAB T'TST_NO_HLT,APT_ERR_PC ; Save the error PC
                       114   MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       115   BSBW APT_COMMAND  ; Call APT routine
                       116   BRB T'TST_NO_S1   ; Continue
                       117 T'TST_NO'_HLT:
                       118   HALT    ; ERROR!!! Program out of sequence
                       119 T_TST_NO'_S1:
                       120   MOVAL T_TST_NO_S1,RET_PC  ; Store return PC
                       121   .ENDM SEQ
                       122
                       123
                       124
                       125 ;
                       126 ; EQUATED SYMBOLS:
                       127 ;
```

```
                    0000   128
00000003            0000   129 PRIMARY_REV == 3
00000003            0000   130 SECONDARY_REV == 3
                    0000   131
0000001F            0000   132 TB_VAL  == 31  ; TB Valid Bit, PTE<31>
03FF8000            0000   133 PTE_MASK ==  X3FF8000 ; Mask for PTE<25:15>, (indeterminate)
                    0000   134
0000FE00            0000   135 APT_CTRL_FLAGS ==  X0000FE00
0000FE04            0000   136 APT_CMD_WORD ==  X0000FE04
0000FE50            0000   137 APT_TEST_NUM ==  X0000FE50 ; Address for absolute test number
0000FE52            0000   138 APT_TEST_COUNT ==  X0000FE52 ; Address for test counter
0000FE44            0000   139 APT_ERR_PC ==  X0000FE44 ; Address for PC at time of error
0000FE40            0000   140 APT_ERR_FLAG ==  X0000FE40 ; Address for error flag
0000FE40            0000   141 APT_MSG_TYPE ==  X0000FE40
0000FE54            0000   142 APT_PASS_CNT ==  X0000FE54 ; Address for pass count
                    0000   143
00010300            0000   144 SPTE  ==  X10300  ; Address for PTE in SPT
00010000            0000   145 SYS_PTBL ==  X10000  ; Address for System Page Table
80011000            0000   146 P0_PTBL  ==  X80011000 ; Address for P0 Page Table
80012000            0000   147 P1_PTBL  ==  X80012000 ; Address for P1 Page Table
00011300            0000   148 P0PTE_VPN1 ==  X11300  ; Address for PTE test 35
00011304            0000   149 P0PTE_VPN2 ==  X11304  ; Address for PTE test 35
000181FE            0000   150 END_VPN1 ==  X181FE  ; Byte 511 of VPN1
                    0000   151
00FFF460            0000   152 SYNC_POINT ==  XFFF460 ; This UET Address is referenced
                    0000   153       ; in scope loops where a Cache Flush
                    0000   154       ; is performed. Unibus Master Sync
                    0000   155       ; can therefore be used as a Sync Point
                    0000   156       ; available at slot 4 pin C90 of UBI
                    0000   157       ; module.
                    0000   158
00FFF460            0000   159 UET_ADDR ==  XFFF460 ; UET Address Register <15:00>
00FFF464            0000   160 UET_CNTL ==  XFFF464 ; UET Control Register
00F30800            0000   161 UNI_MAP  ==  XF30800 ; Unibus Map Register
                    0000   162
00000007            0000   163 READY  == 7  ; Ready Bit #7
                    0000   164
00000025            0000   165 CADR  == 37  ; Cache Disable Register
00000027            0000   166 CAER  == 39  ; Cache Error Register
00000012            0000   167 IPL  == 18  ; Interrupt Priority Level
00000037            0000   168 IO_RESET == 55  ; Initialize Unibus
00000000            0000   169 KSP  == 00  ; Kernal Stack Pointer
00000004            0000   170 ISP  == 04  ; INTERRUPT STACK POINTER
00000020            0000   171 RXCS  == 32  ; RECEIVER STATUS REGISTER
00000021            0000   172 RXDB  == 33  ; RECEIVER DATA BUFFER
00000026            0000   173 MCESR  == 38  ; Machine Check Error Summary Reg.
00000038            0000   174 MME  == 56  ; Memory Management Enable
00000008            0000   175 P0BR  == 08  ; P0 Base Register, (also used to load
                    0000   176       ; VA when using TB Data IPR)
00000009            0000   177 P0LR  == 09  ; P0 Length Register
0000000A            0000   178 P1BR  == 10  ; P1 Base Register
0000000B            0000   179 P1LR  == 11  ; P1 Length Register
0000000C            0000   180 SBR  == 12  ; System Base Register
00000011            0000   181 SCBB  == 17  ; System Control Block Base
0000000D            0000   182 SLR  == 13  ; System Length Register
0000003B            0000   183 TB_DATA  == 59  ; Translation Buffer Data
00000024            0000   184 TBDR  == 36  ; Tranlation Buffer Disable Reg.
```

```
00000039  0000   185 TBIA  == 57  ; Translation Buffer Invalidate All
0000003A  0000   186 TBIS  == 58  ; Translation Buffer Invalidate Single
00000022  0000   187 TXCS  == 34  ; Console Trans. Status Register
00000023  0000   188 TXDB  == 35  ; Console Trans. Data buffer
          0000   189
          0000   190
```

```
0000    192 ;**
0000    193 ; MACHINE-CHECK LOGOUT STACK:
0000    194 ;
0000    195 ; This is the format of the stack after machine-check
0000    196 ; exceptions.
0000    197 ;
0000    198 ;  ---------------------------------------------;
0000    199 ;      Length Parameter  (28 hex)       (1)
0000    200 ;  ---------------------------------------------
0000    201 ;      Error Code         (2)
0000    202 ;  ---------------------------------------------
0000    203 ;      VA Register
0000    204 ;  ---------------------------------------------
0000    205 ;      PC At Time Of Error
0000    206 ;  ---------------------------------------------
0000    207 ;      MDR          (6)
0000    208 ;  ---------------------------------------------
0000    209 ;      Saved Mode Register       (3)
0000    210 ;  ---------------------------------------------
0000    211 ;      Read Lock Timeout        (3) (5)
0000    212 ;  ---------------------------------------------
0000    213 ;      TB Group Parity Error Register      (3)
0000    214 ;  ---------------------------------------------
0000    215 ;      Cache Error Register (CAER)      (3)
0000    216 ;  ---------------------------------------------
0000    217 ;      Bus Error Register        (3)
0000    218 ;  ---------------------------------------------
0000    219 ;      Machine Check Error       (3)
0000    220 ;      Summary Register     (MCESR)
0000    221 ;  ---------------------------------------------
0000    222 ;      PC
0000    223 ;  ---------------------------------------------
0000    224 ;      PSL
0000    225 ;  ---------------------------------------------
0000    226 ;
0000    227 ;
0000    228 ;  (1) Length Parameter:
0000    229 ;
0000    230 ; Contains the number of bytes pushed on the stack excluding
0000    231 ; the length parameter, PC, and PSL.
0000    232 ;
0000    233 ;  (2) Error Code:
0000    234 ;
0000    235 ; This code will distinguish the following conditions:
0000    236 ;
0000    237 ; code  condition
0000    238 ;
0000    239 ; 1  Control Store Parity Error (Abort)
0000    240 ; 2  TB Error, Bus Error, or Unaligned Unibus
0000    241 ;    Reference (Abort)
0000    242 ; 3  Not used
0000    243 ; 4  Not used
0000    244 ; 5  Corrected Memory Data (Interrupt IPL = 1A)
0000    245 ; 6  "Not Supposed to be here" (Abort) (for unused
0000    246 ;    C/S locations)
0000    247 ; 7  "Unused IRD ROM Slot"
0000    248 ;
```

```
                     0000   249 ;  (3) The 4-bit Memory Status and Control Registers will be zero
                     0000   250 ; extended to a long word.  (Consult CMI spec.)
                     0000   251 ;
                     0000   252 ;  (4) The contents of WDR are only meaningful if this is a
                     0000   253 ; micro-trap detected error that occurred during a
                     0000   254 ; micro-instruction that was loading WDR.
                     0000   255 ;
                     0000   256 ;  (5) This is called "Write Vector Occured Register"
                     0000   257 ; (Consult CMI spec.)
                     0000   258 ;
                     0000   259 ;  (6) The Memory Data Register will contain the 'bad data' as a
                     0000   260 ; result of a Cache Data Memory Parity Error.
                     0000   261 ;
                     0000   262 ;
                     0000   263
                     0000   264
                     0000   265
                     0000   266
                     0000   267
                     0000   268
                     0000   269
                     0000   270 ;
                     0000   271 ;**
                     0000   272 ;  Definitions for reading Machine Check Logout Stack
                     0000   273 ;--
                     0000   274
00000024             0000   275 BUS_ER_REG ==  X24  ; Displacement for reading contents of
                     0000   276       ; Bus Error Register on Machine Check
                     0000   277       ; Logout Stack
00000020             0000   278 CA_ER_REG ==  X20  ; Displacement for reading contents of
                     0000   279       ; Cache Error Register on Machine Check
                     0000   280       ; Logout Stack
0000000E             0000   281 CA_ER_BITS ==  XE  ; Bits<3:1> of CAER
00000028             0000   282 MA_CHK_REG ==  X28  ; Displacement for reading contents of
                     0000   283       ; Machine Check Register on Machine
                     0000   284       ; Check Logout Stack
00000003             0000   285 RD_NXM  == 3  ; Bit<3> of Bus Error Register
00000002             0000   286 TB_ER_BIT == 2  ; Bit<2> of MCESR
                     C000   287
                     0000   288
                     0000   289 ;**
                     0000   290 ; The first action of the code is to disable the Cache and TB.
                     0000   291 ; If unsuccessful, manually disable the Cache by depositing
                     0000   292 ; a 1 into the CADR (IPR 25) by use of the console. To manually
                     0000   293 ; disable the TB deposit a 3 into the TBDR (IPR 24).
                     0000   294 ;--
                     0000   295
00000200             0000   296  . =  X200   ; Start program at address 200
                     0200   297
085B   31             0200   298  BRW INIT   ; Start the test program
                     0203   299
00000208             0203   300  . =  X208
00000224             0208   301  .ADDRESS PROGRAM_NAME
00000003             020C   302  .LONG  PRIMARY_REV
00000003             0210   303  .LONG  SECONDARY_REV
                     0214   304
00000220             0214   305  . =  X220   ; Starting address for special
```

```
                     0220   306      ; TB trouble shooting test
                     0220   307
              0818    31     0220   308  BRW SPEC_TB_TEST  ; Start Special TB Test
                     0223   309
                     0223   310 ;**
                     0223   311 ;  Definitions for reading Machine Check Logout Stack
                     0223   312 ;--
                     0223   313
          00000024   0223   314 BUS_ER_REG ==  X24  ; Displacement for reading contents of
                     0223   315      ; Bus Error Register on Machine Check
                     0223   316      ; Logout Stack
          00000020   0223   317 CA_ER_REG ==  X20  ; Displacement for reading contents of
                     0223   318      ; Cache Error Register on Machine Check
                     0223   319      ; Logout Stack
          0000000E   0223   320 CA_ER_BITS ==  XE  ; Bits<3:1> of CAER
          00000028   0223   321 MA_CHK_REG ==  X28  ; Displacement for reading contents of
                     0223   322      ; Machine Check Register on Machine
                     0223   323      ; Check Logout Stack
          00000003   0223   324 RD_NXM  =- 3  ; Bit<3> of Bus Error Register
          00000002   0223   325 TB_ER_BIT == 2  ; Bit<2> of MCESR
                     0223   326
                     0223   327
                     0223   328
                     0223   329 ;
                     0223   330 ; OWN STORAGE:
                     0223   331 ;
                     0223   332
          00000001   0223   333 BIT0  =  X00000001
          00000002   0223   334 BIT1  =  X00000002
          00000004   0223   335 BIT2  =  X00000004
          00000008   0223   336 BIT3  =  X00000008
          00000010   0223   337 BIT4  =  X00000010
          00000020   0223   338 BIT5  =  X00000020
          00000040   0223   339 BIT6  =  X00000040
          00000080   0223   340 BIT7  =  X00000080
          00000100   0223   341 BIT8  =  X00000100
          00000200   0223   342 BIT9  =  X00000200
          00000400   0223   343 BIT10  =  X00000400
          00000800   0223   344 BIT11  =  X00000800
          00001000   0223   345 BIT12  =  X00001000
          00002000   0223   346 BIT13  =  X00002000
          00004000   0223   347 BIT14  =  X00004000
          00008000   0223   348 BIT15  =  X00008000
          00010000   0223   349 BIT16  =  X00010000
          00020000   0223   350 BIT17  =  X00020000
          00040000   0223   351 BIT18  =  X00040000
          00080000   0223   352 BIT19  =  X00080000
          00100000   0223   353 BIT20  =  X00100000
          00200000   0223   354 BIT21  =  X00200000
          00400000   0223   355 BIT22  =  X00400000
          00800000   0223   356 BIT23  =  X00800000
          01000000   0223   357 BIT24  =  X01000000
          02000000   0223   358 BIT25  =  X02000000
          04000000   0223   359 BIT26  =  X04000000
          08000000   0223   360 BIT27  =  X08000000
          10000000   0223   361 BIT28  =  X10000000
          20000000   0223   362 BIT29  =  X20000000
```

```
40000000  0223   363 BIT30  =  X40000000
80000000  0223   364 BIT31  =  X80000000
          0223   365
          0223   366 ;
          0223   367 ; EQUATED SYMBOLS:
          0223   368 ;
          0223   369
00F08000  0223   370 PCS_PTH_MIC_ADDR =  X00F08000
00F00000  0223   371 PCS_PTH_BIT_ADDR =  X00F00000
00F0C000  0223   372 PCS_SELECT_REG  =  X00F0C000
          0223   373
00000000  0223   374 PCS_ENA_WRT_PCS  =  X00000000
00000001  0223   375 PCS_ENA_WRT_PTH_BIT  =  X00000001
          0223   376
00002000  0223   377 PCS_PTH_BIT_SIZE_BITS =  X00002000
00000100  0223   378 PCS_PTH_BIT_SIZE_LWORDS =  X00000100
00001000  0223   379 PCS_PTH_MIC_SIZE_LWORDS =  X00001000
00000400  0223   380 PCS_PTH_MIC_SIZE_MWORDS =  X00000400
          0223   381
000AAAAA  0223   382 PCS_TEST_PATTERN =  X000AAAAA
          0223   383
00000000  0223   384 KSP   =  X00000000
00000004  0223   385 ISP   =  X00000004
00000011  0223   386 SCBB   =  X00000011
00000012  0223   387 IPL   =  X00000012
00000020  0223   388 RXCS   =  X00000020
00000021  0223   389 RXDB   =  X00000021
00000022  0223   390 TXCS   =  X00000022
00000023  0223   391 TXDB   =  X00000023
00000018  0223   392 ICCS   =  X00000018
00000019  0223   393 NICR   =  X00000019
0000001A  0223   394 ICR   =  X0000001A
0000001D  0223   395 TODR   =  X0000001D
          0223   396  :
          0223   397  ; The following Byte 'PCS_STATUS_FLAGS' is used as flag bit to
          0223   398  ; signify what has been established within the loader program.  The
          0223   399  ; usage of these bits is as follows ( a value of 1 is the true state
          0223   400  ; of the flag):
          0223   401  :
          0223   402  ;  Bit 0 = PCS is Present
          0223   403  ;  Bit 1 = Patch bits are loaded
          0223   404  ;  Bit 2 = PCS microwords are loaded
          0223   405  ;  Bit 3 =
          0223   406  ;  Bit 4 =
          0223   407  ;  Bit 5 =
          0223   408  ;  Bit 6 =
          0223   409  ;  Bit 7 =
          0223   410 PCS_STATUS_FLAGS::
       00 0223   411  .BYTE 0
```

```
                                        0224   412  .SBTTL Messages used by this program
                                        0224   413 ;++
                                        0224   414 ; The folowing are messages used by this program.
                                        0224   415 ;--
                                        0224   416
41 56 20 2D 2D 20 4C 41 4B 43 45 00     0224   417 PROGRAM_NAME: .ASCIC  ECKAL -- VAX 11/750 Cache/TB Diagnostic
63 61 43 20 30 35 37 2F 31 31 20 58     0230
6F 6E 67 61 69 44 20 42 54 2F 65 68     023C
                     63 69 74 73       0248
                                 27     0224
61 50 20 66 6F 20 64 6E 45 0A 0D 00'    024C   418 ENDPASS:: .ASCIC <13><10> End of Pass, ECKAL V3.4 <7> ; Endpass message
33 56 20 4C 41 4B 43 45 20 2C 73 73     0258
                     07 34 2E 34       0264
                              1A       024C
                  20 3E 53 44 00'      0267   419 PROMPT: .ASCIC  DS>
                              04       0267
4D 20 6E 69 20 53 43 50 20 6F 4E 00'    026C   420 NO_PCS: .ASCIC  N  PCS in Module (L0008) in this 11/750
38 30 30 30 4C 28 20 65 6C 75 64 6F     0278
31 31 20 73 69 68 74 20 6E 69 20 29     0284
                     30 35 37 2F 34     0290
                                 27     026C
                                        0294   421
                  00000000              0294   422 PCS_SAVE_COUNT:: .LONG 0
                  00000000              0298   423 PCS_INIT_SAVE::  .LONG 0
```

```
                                  029C    424  .SBTTL Own Storage
                                  029C    425 ;
                                  029C    426 ; OWN STORAGE:
                                  029C    427 ;
                                  029C    428
                                  029C    429
00000010 00000008 00000004 00000000  029C    430 CA_INDX_VAL:: .LONG 0,4,8, X10 ; Cache Index value for test 06,07
00000100 00000080 00000040 00000020  02AC    431   .LONG  X20, X40, X80, X100
         00000800 00000400 00000200  02BC    432   .LONG  X200, X400, X800
                           00AAAAA8  02C8    433 CA_TST_ADR:: .LONG  XAAAAA8 ; Cache Test Address1 for Test 13
                           00D55554  02CC    434   .LONG  XD55554 ; Cache Test Address2 for Test 13
                           00CCCCCC  02D0    435   .LONG  XCCCCCC ; Cache Test Address3 for Test 13
                           00BC3C3C  02D4    436   .LONG  XBC3C3C ; Cache Test Address4 for Test 13
                           00FC03FC  02D8    437   .LONG  XFC03FC ; Cache Test Address5 for Test 13
80005E1E 80006666 80006AAA 80005555  02DC    438 UNI_PFN:: .LONG  X80005555, X80006AAA, X80006666, X80005E1E
                           80007E01  02EC    439   .LONG  X80007E01 ; Unibus Map PFNs for Test 13
                           55550000  02F0    440 TB_PAT_A:: .LONG  X55550000 ; TB pattern1 for Tests 16,17
                           2AAA0000  02F4    441   .LONG  X2AAA0000 ; TB pattern2 for Tests 16,17
                           33330000  02F8    442   .LONG  X33330000 ; TB pattern3 for Tests 16,17
                           0F0F0000  02FC    443   .LONG  X0F0F0000 ; TB pattern4 for Tests 16,17
                           00FF0000  0300    444   .LONG  X00FF0000 ; TB pattern5 for Tests 16,17
                           D4005555  0304    445 TB_PAT_B:: .LONG  XD4005555 ; TB pattern1 for Tests 18,19
                           A8002AAA  0308    446   .LONG  XA8002AAA ; TB pattern2 for Tests 18,19
                           CC003333  030C    447   .LONG  XCC003333 ; TB pattern3 for Tests 18,19
                           BC000F0F  0310    448   .LONG  XBC000F0F ; TB pattern4 for Tests 18,19
                           FC0000FF  0314    449   .LONG  XFC0000FF ; TB pattern5 for Tests 18,19
                           00000000  0318    450 TB_PAT_C:: .LONG 0  ; TB pattern1 for Tests 22-25
                           00000200  031C    451   .LONG  X200  ; TB pattern2 for Tests 22-25
                           00000400  0320    452   .LONG  X400  ; TB pattern3 for Tests 22-25
                           00000800  0324    453   .LONG  X800  ; TB pattern4 for Tests 22-25
                           00001000  0328    454   .LONG  X1000  ; TB pattern5 for Tests 22-25
                           00002000  032C    455   .LONG  X2000  ; TB pattern6 for Tests 22-25
                           00004000  0330    456   .LONG  X4000  ; TB pattern7 for Tests 22-25
                           00008000  0334    457   .LONG  X8000  ; TB pattern8 for Tests 22-25
                           80000000  0338    458   .LONG  X80000000 ; TB pattern9 for Tests 22-25
                           AAAAAAAA  033C    459 TB_PAT_D:: .LONG  XAAAAAAAA ; TB pattern1 for Test 33
                           55555555  0340    460   .LONG  X55555555 ; TB pattern2 for Test 33
                           F0F0F0F0  0344    461   .LONG  XF0F0F0F0 ; TB pattern3 for Test 33
                           00018000  0348    462 VA1::  .LONG  X18000  ; TB VA for Mapping Tests
                           80018000  034C    463 VA2::  .LONG  X80018000 ; TB VA for Mapping Tests
80018000 00018000 40018000 80018000  0350    464 VA3::  .LONG  X40018000, X18000, X80018000
         40018000 00018000  035C    465   .LONG  X18000, X40018000 ; TB VAs for Mapping Tests
                            0364    466
                            0364    467
                            0364    468   .ALIGN PAGE  ; Align SCB on page boundary
                            0400    469
                            0400    470
                            0400    471 SCB::     ; SYSTEM CONTROL BLOCK
                           00000000 0400    472 SCB0::  .LONG 0  ; (RESERVED)
                           00000000 0404    473 SCB04::  .LONG 0  ; MACHINE CHECK
                           00000420 0408    474 SCB8::  .BLKL 6  ;
                           00000000 0420    475 SCB20::  .LONG 0  ; ACCESS CONTROL VIOLATION
                           00000460 0424    476 SCB24::  .BLKL 15  ;
                           00000000 0460    477 SCB60::  .LONG 0  ; WRITE NON-EXISTENT MEMORY
                           00000600 0464    478 SCB64::  .BLKL 103  ;
                                    0600    479
                                    0600    480
```

```
                     0600    481 TCB::     ; TRAP CATCHER BLOCK
00000000             0600    482 TCB0::  .LONG 0  ; (RESERVED)
00000000             0604    483 TCB4::  .LONG 0  ; Service Routine exists for MACH. CHK
00000000             0608    484 TCB8::  .LONG 0  ; KERNEL STACK NOT VALID
00000000             060C    485 TCBC::  .LONG 0  ; POWER FAIL
00000000             0610    486 TCB10:: .LONG 0  ; RESERVED/PRIVILEGED INSTR.
00000000             0614    487 TCB14:: .LONG 0  ; CUSTOMER RESERVED INSTRUCTION
00000000             0618    488 TCB18:: .LONG 0  ; RESERVED OPERAND
00000000             061C    489 TCB1C:: .LONG 0  ; RESERVED ADDRESSING MODE
00000000             0620    490 TCB20:: .LONG 0  ; Service Routine exists for ACV's
00000000             0624    491 TCB24:: .LONG 0  ; TRANSLATION NOT VALID
00000000             0628    492 TCB28:: .LONG 0  ; TRACE PENDING
00000000             062C    493 TCB2C:: .LONG 0  ; BREAKPOINT INSTRUCTION
00000000             0630    494 TCB30:: .LONG 0  ; COMPATIBILITY MODE
00000000             0634    495 TCB34:: .LONG 0  ; ARITHMETIC TRAP
00000000             0638    496 TCB38:: .LONG 0  ; UNUSED
00000000             063C    497 TCB3C:: .LONG 0  ; UNUSED
00000000             0640    498 TCB40:: .LONG 0  ; CHANGE MODE TO KERNEL
00000000             0644    499 TCB44:: .LONG 0  ; CHANGE MODE TO EXECUTIVE
00000000             0648    500 TCB48:: .LONG 0  ; CHANGE MODE TO SUPERVISOR
00000000             064C    501 TCB4C:: .LONG 0  ; CHANGE MODE TO USER
00000684             0650    502 TCB50:: .BLKL 13  ; UNUSED
                     0684    503       ; Service Routine exists for TCB60,
                     0684    504       ;      WRITE NXM INTERRUPT
00000000             0684    505 TCB84:: .LONG 0  ; SOFTWARE LEVEL 1
00000000             0688    506 TCB88:: .LONG 0  ; SOFTWARE LEVEL 2
00000000             068C    507 TCB8C:: .LONG 0  ; SOFTWARE LEVEL 3
00000000             0690    508 TCB90:: .LONG 0  ; SOFTWARE LEVEL 4
00000000             0694    509 TCB94:: .LONG 0  ; SOFTWARE LEVEL 5
00000000             0698    510 TCB98:: .LONG 0  ; SOFTWARE LEVEL 6
00000000             069C    511 TCB9C:: .LONG 0  ; SOFTWARE LEVEL 7
00000000             06A0    512 TCBA0:: .LONG 0  ; SOFTWARE LEVEL 8
00000000             06A4    513 TCBA4:: .LONG 0  ; SOFTWARE LEVEL 9
00000000             06A8    514 TCBA8:: .LONG 0  ; SOFTWARE LEVEL A
00000000             06AC    515 TCBAC:: .LONG 0  ; SOFTWARE LEVEL B
00000000             06B0    516 TCBB0:: .LONG 0  ; SOFTWARE LEVEL C
00000000             06B4    517 TCBB4:: .LONG 0  ; SOFTWARE LEVEL D
00000000             06B8    518 TCBB8:: .LONG 0  ; SOFTWARE LEVEL E
00000000             06BC    519 TCBBC:: .LONG 0  ; SOFTWARE LEVEL F
00000000             06C0    520 TCBC0:: .LONG 0  ; INTERVAL TIMER
000006F8             06C4    521 TCBC4:: .BLKL 13  ; UNUSED
00000000             06F8    522 TCBF8:: .LONG 0  ; CONSOLE TERMINAL RECIEVER
00000000             06FC    523 TCBFC:: .LONG 0  ; CONSOLE TERMINAL TRANSMITTER
00000800             0700    524 TCB100:: .BLKL 64  ; DEVICE VECTORS
                     0800    525
                     0800    526
00000A00             0800    527 BOT_OF_STACK:: .BLKL 128  ; Allocate a stack area
00000000             0A00    528 TOP_OF_STACK:: .LONG 0  ; Set top of stack pointer
                     0A04    529
                     0A04    530
                     0A04    531
                00   0A04    532 NXM_WT_FLAG:: .BYTE 0  ; Non-Existent Memory Write Flag
                00   0A05    533 NXM_RD_FLAG:: .BYTE 0  ; Non-Existent Memory Read Flag
                00   0A06    534 ACC_VIO_FLAG:: .BYTE 0  ; Access Control Violation Flag
                     0A07    535
00000000             0A07    536 RET_PC:: .LONG 0  ; Modified PC Address for looping
                     0A0B    537       ; on Machine Check Exceptions.
```

```
00000000  0A0B   538 CONT_PC:: .LONG 0  ; Modified PC Address for continuing
          0A0F   539      ; test on expected Machine Checks.
          0A0F   540
          0A0F   541
          0A0F   542
```

```
                                0A0F    544
                                0A0F    545 ;**
                                0A0F    546 ;
                                0A0F    547 ; SUBROUTINE INC_INDEX
                                0A0F    548 ;
                                0A0F    549 ;  This subroutine is used to generate 256 Indexing addresses
                                0A0F    550 ;  needed for the Dynamic Testing of the TB Tag and Data Store
                                0A0F    551 ;  Rams(tests 26-31). The addresses start from 00000000 to
                                0A0F    552 ;  0000FE00 (Process half), and proceed from 80000000 to
                                0A0F    553 ;  to 8000FE00 (System half).
                                0A0F    554 ;
                                0A0F    555 ;--
                                0A0F    556
                                0A0F    557
                                0A0F    558 INC_INDEX::     ; Continued on next line
     50    00000200 8F    C0  0A0F    559  ADDL2 #X200,R0   ; Generate next Index value
           07 50    10    E1  0A16    560  BBC #16,R0,10$   ; Branch if Process half of Index val.
     50    80000000 8F    D0  0A1A    561  MOVL #X80000000,R0   ; Set up for first System Index value
              08    50    DA  0A21    562 10$: MTPR R0,#PQBR   ; Load VA
                          05  0A24    563  RSB     ; Return
                                0A25    564
                                0A25    565
                                0A25    566
                                0A25    567
                                0A25    568 ;**
                                0A25    569 ;
                                0A25    570 ; SUBROUTINE DEC_INDEX
                                0A25    571 ;
                                0A25    572 ;  This subroutine is similar to subroutine INC_INDEX except
                                0A25    573 ;  that its Index addresses  are generated starting at the
                                0A25    574 ;  the highest and proceeding to the lowest, from 8000FE00 to
                                0A25    575 ;  80000000, and from 0000FE00 to 00000000.
                                0A25    576 ;
                                0A25    577 ;--
                                0A25    578
                                0A25    579
                                0A25    580 DEC_INDEX::     ; Continued on next line
     50    00000200 8F    C2  0A25    581  SUBL2 #X200,R0   ; Generate next Index value
           07 50    1E    E1  0A2C    582  BBC #30,R0,10$   ; Branch if System half of Index val.
     50    0000FE00 8F    D0  0A30    583  MOVL #XFE00,R0   ; Set up for first Process Index value
              08    50    DA  0A37    584 10$: MTPR R0,#PQBR   ; Load VA
                          05  0A3A    585  RSB     ; Return
```

```
                        0A3B   586  .SBTTL Special TB Trouble-Shooting test
                        0A3B   587 ;++
                        0A3B   588 ;
                        0A3B   589 ; SPECIAL TB TROUBLE-SHOOTING TEST
                        0A3B   590 ;
                        0A3B   591 ; This routine can be used to write, read, and loop
                        0A3B   592 ; on any desired VA and PTE.
                        0A3B   593 ;
                        0A3B   594 ; Deposit desired VA into R5
                        0A3B   595 ; Deposit desired PTE into R1
                        0A3B   596 ; Deposit desired replacement code into R6 as follows:
                        0A3B   597 ;
                        0A3B   598 ;   hex A = replace Group 0, disable Group 1
                        0A3B   599 ;   hex D = replace Group 1, disable Group 0
                        0A3B   600 ;        0 = enable Random Replacement
                        0A3B   601 ;
                        0A3B   602 ; Start test at address 220, routine will continually loop
                        0A3B   603 ; on test unless BRB instruction is replaced with NOP.
                        0A3B   604 ;
                        0A3B   605 ; Six NOP instructions are also provided to modify
                        0A3B   606 ; the routine.
                        0A3B   607 ;
                        0A3B   608 ;--
                        0A3B   609
                        0A3B   610 SPEC_TB_TEST::
            3A   55  DA 0A3B   611 10$: MTPR R5,#TBIS  ; Invalidate TB VA under test
            08   55  DA 0A3E   612  MTPR R5,#P0BR  ; Load desired VA
            24   56  DA 0A41   613  MTPR R6,#TBDR  ; Set desired replacement
            3B   51  DA 0A44   614  MTPR R1,#TB_DATA  ; Write PTE into TB
            52   3B  DB 0A47   615  MFPR #TB_DATA,R2  ; Read PTE into R2
52   03FF8000 8F  CA 0A4A   616  BICL #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
    53   52  51  CD 0A51   617  XORL3 R1,R2,R3  ; R1 contains PTE written
                        0A55   618       ; R2 contains PTE read
                        0A55   619       ; R3 contains PTE XOR
                        0A55   620       ; R5 contains VA
                        0A55   621       ; R6 contans replacement code
                    01  0A55   622  NOP     ;
                    01  0A56   623  NOP     ;
                    01  0A57   624  NOP     ;
                    01  0A58   625  NOP     ;
                    01  0A59   626  NOP     ;
                    01  0A5A   627  NOP     ;
                DE  11  0A5B   628  BRB 10$   ; Loop on test
                    00  0A5D   629  HALT    ; Halt test
```

```
                                0A5E   630  .SBTTL  INITIALIZATION ROUTINE
                                0A5E   631 ;++
                                0A5E   632 ; FUNCTIONAL DESCRIPTION:
                                0A5E   633 ;
                                0A5E   634 ; This routine sets up the SCB to point to trap catchers, and
                                0A5E   635 ; exception and interrupt routines. The stack pointer is also
                                0A5E   636 ; initialized.
                                0A5E   637 ; The Machine Check, Access Control Violation, and Write NXM
                                0A5E   638 ; service routines are also contained here.
                                0A5E   639 ;
                                0A5E   640 ;
                                0A5E   641 ;
                                0A5E   642 ;
                                0A5E   643 ;
                                0A5E   644 ;--
              25   01    DA   0A5E   645 INIT: MTPR #1,#CADR  ; Disable the Cache
        5A  00000000'EF    9E   0A61   646  MOVAB 0,R10   ; PUT LOAD ADDRESS OF PROGRAM IN R10
            0000FE54'EF    7C   0A68   647  CLRQ APT_PASS_CNT  ; Initialize pass count
            0000FE40'EF    D4   0A6E   648  CLRL APT_MSG_TYPE  ; Clear the message type
                           0A74   649 ;
                           0A74   650 ; check for APT load
                           0A74   651 ;
     13 0000FE00'EF    1F    E1   0A74   652  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT
           5E   81 AF    DE   0A7C   653  MOVAL TOP_OF_STACK,SP  ; SP Points to the top of stack
              00   5E    DA   0A80   654  MTPR SP,#KSP   ; Load the KSP with the top of stack
              04   5E    DA   0A83   655  MTPR SP,#ISP   ; Load the ISP
                 02B3    30   0A86   656  BSBW TYPE_PROMPT  ; Type the APT prompt
                 0237    30   0A89   657  BSBW CHECK_APT_CMD  ; Check the APT command word
                 48F5    31   0A8C   658  BRW PCS_INIT  ; Go to the PCS init routine
            0000FE00 EF    D4   0A8F   659 10$: CLRL APT_CTRL_FLAGS  ; Clear the APT control flags
          5E   FF67 CF    DE   0A95   660  MOVAL TOP_OF_STACK,SP  ; SP Points to the top of stack
               51   01    D0   0A9A   661  MOVL #1,R1   ; Init index
          53   F783 CF    9A   0A9D   662  MOVZBL PROGRAM_NAME,R3  ; Get character count
        52   F77D CF41    9A   0AA2   663 20$: MOVZBL PROGRAM_NAME R1 ,R2 ; Get character
                 0286    30   0AA8   664  BSBW PRINT_CHARACTER  ; Print it
             F3 51   53    F3   0AAB   665  AOBLEQ R3,R1,20$  ; Print them all
                 48D2    31   0AAF   666  BRW PCS_INIT  ; Go to PCS init routine
```

```
                5E    FF4A CF    DE   0AB2   667 START: MOVAL TOP_OF_STACK,SP  ; SP Points to the top of stack
                      00   5E    DA   0AB7   668  MTPR SP,#KSP   ; Load the KSP with the top of stack
                      53   01    D0   0ABA   669  MOVL #1,R3    ; Initialize counter
                50    FB3F CF    DE   0ABD   670  MOVAL TCB,R0   ; Load Trap Catcher Block address
                51    F93A CF    DE   0AC2   671  MOVAL SCB,R1   ; Load System Control Block address
                      11   51    DA   0AC7   672  MTPR R1,#SCBB  ; Set the SCBB to point to base addr.
                      81   80    DE   0ACA   673 10$: MOVAL (R0)+,(R1)+  ; Load the SCB with trap catchers
              F5 53    0000007F 8F    F3   0ACD   674  AOBLEQ #127,R3,10$  ; Continue until the SCB is full
           F926 CF    00000B18'EF    DE   0AD5   675  MOVAL MACH_CHK_SRV,SCB04 ; Load Mach. Chk. Vector address
           F939 CF    00000C14'EF    DE   0ADE   676  MOVAL ACV_SRV,SCB20  ; Load ACV Vector address
           F970 CF    00000BCC'EF    DE   0AE7   677  MOVAL WT_NXM_SRV,SCB60 ; Load Wt. NXM Vector address
                      0000FE52 EF    B4   0AF0   678  CLRW APT_TEST_COUNT  ; INIT TEST COUNTER
                      0000FE40'EF    94   0AF6   679  CLRB APT_ERR_FLAG  ; Clear the Error flag
                          FF04 CF    94   0AFC   680  CLRB NXM_WT_FLAG  ; Clear flag
                          FF01 CF    94   0B00   681  CLRB NXM_RD_FLAG  ; Clear flag
                          FEFE CF    94   0B04   682  CLRB ACC_VIO_FLAG  ; Clear flag
                          12   00    DA   0B08   683  MTPR #0,#IPL   ; Lower IPL
                               00    DD   0B0B   684  PUSHL #0   ; Push zero for PSL on Stack
                      00000D70 EF    DF   0B0D   685  PUSHAL TEST_1   ; Push Test 1 Address on Stack
                               02    02   0B13   686  REI    ; Pop the Stack
                               00    00   0B14   687  HALT    ; SHOULD NEVER GET HERE
                                  00000001   0B15   688  $TN = 1
```

```
                                0B15    689  .SBTTL Machine Check Service Routine
                                0B15    690 ;++
                                0B15    691 ; Machine Check Service Routine:
                                0B15    692 ;
                                0B15    693 ; This routine is set up to handle all Cache, TB, and Read
                                0B15    694 ; NXM Machine Check Exceptions.  Cache and TB errors will
                                0B15    695 ; cause a halt to occur. Consult the Machine Check Logout
                                0B15    696 ; Stack for location of error information on the stack.
                                0B15    697 ; In the case where it is desirable to loop on error, the
                                0B15    698 ; error halt should be replaced by a NOP instruction, then
                                0B15    699 ; the Stack Pointer will be reset and the failing test will
                                0B15    700 ; be repeated.
                                0B15    701 ;
                                0B15    702 ; Cache Test 14 (Locked Read Test) will make use of the Read
                                0B15    703 ; NXM exception in order to test that the read portion of a
                                0B15    704 ; Locked Read instruction does not get cached. This will be
                                0B15    705 ; accomplished by setting a NXM_RD_FLAG at the start of the
                                0B15    706 ; test(this flag will be cleared for all other tests where the
                                0B15    707 ; occurence of a Read NXM will be an unexpected error).
                                0B15    708 ;
                                0B15    709 ;
                                0B15    710 ;
                                0B15    711 ;
                                0B15    712 ;--
                                0B15    713  .ALIGN LONG
                                0B18    714 MACH_CHK_SRV:    ;
          25    01    DA      0B18    715  MTPR #1,#CADR   ; Disable Cache
        24 AE   03    E1      0B1B    716  BBC #RD_NXM,BUS_ER_REG(SP),-;
                      38      0B1F    717   20$    ; Check for Read NXM Exception
     15 FEE0 CF   00    E1      0B20    718  BBC #0,NXM_RD_FLAG,10$ ; Check status of NXM_RD_FLAG
              26   0F    DA      0B26    719  MTPR #^XF,#MCESR   ; Clear MCESR
              27   0F    DA      0B29    720  MTPR #^XF,#CAER   ; Clear CAER
              FED5 CF    94      0B2C    721  CLRB NXM_RD_FLAG   ; Clear flag
              5E   8E    C0      0B30    722  ADDL2 (SP)+,SP   ; Reset the Stack
        6E    FED4 CF    D0      0B33    723  MOVL CONT_PC,(SP)   ; Load Address to continue test
                 008E    31      0B38    724  BRW 60$    ; Return and continue test
12 0000FE00 EF   1F    E1      0B3B    725 10$: BBC #31,APT_CTRL_FLAGS,15$ ; Branch if not APT mode
0000FE44 EF    55 AF    3E      0B43    726  MOVAW B 15$,APT_ERR_PC ; Save Error PC
   0000FE40 EF   01    88      0B4B    727  BISB2 #1,APT_ERR_FLAG   ; Set Error Flag
                012D    30      0B52    728  BSBW APT_COMMAND   ;
                        00      0B55    729 15$: HALT     ; Unexpected Read NXM Exception
                   63   11      0B56    730  BRB 50$    ; Set up to loop on failing test
                   02   E1      0B58    731 20$: BBC #TB_ER_BIT,-   ; Continued on next line
                28 AE           0B5A    732   MA_CHK_REG(SP),- ; Continued again
                   1D           0B5C    733   30$    ; Check for TB error
12 0000FE00 EF   1F    E1      0B5D    734  BBC #31,APT_CTRL_FLAGS,25$ ; Branch if not APT mode
0000FE44 EF    77 AF    3E      0B65    735  MOVAW B 25$,APT_ERR_PC ; Save Error PC
   0000FE40 EF   01    88      0B6D    736  BISB2 #1,APT_ERR_FLAG   ; Set Error Flag
                010B    30      0B74    737  BSBW APT_COMMAND   ;
                        00      0B77    738 25$: HALT     ; TB Error, read Machine Check-
                                0B78    739       ; Logout Stack for information
                   41   11      0B78    740  BRB 50$    ; Set up to loop on failing test
                   0E   93      0B7A    741 30$: BITB #CA_ER_BITS,-   ; Continued on next line
                20 AE           0B7C    742   CA_ER_REG(SP)   ; Check for Cache error
                   1D   13      0B7E    743  BEQL 40$    ; If not Cache error, look for
                                0B80    744       ; other Machine Check Exc.
12 0000FE00 EF   1F    E1      0B80    745  BBC #31,APT_CTRL_FLAGS,35$ ; Branch if not APT mode
```

```
                0000FE44 EF    9A AF    3E  0B88    746  MOVAW B 35$,APT_ERR_PC ; Save Error PC
                   0000FE40'EF    01    88  0B90    747  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                00E8    30  0B97    748  BSBW APT_COMMAND  ;
                                        00  0B9A    749 35$: HALT    ; Cache Error, read Machine Check-
                                            0B9B    750      ; Logout Stack for information
                                     1E     11  0B9B    751  BRB 50$   ; Set up to loop on failing test
                15 0000FE00 EF      1F     E1  0B9D    752 40$: BBC #31,APT_CTRL_FLAGS,45$ ; Branch if not APT mode
             0000FE44 EF    00000BBA EF     3E  0BA5    753  MOVAW 45$,APT_ERR_PC  ; Save Error PC
                   0000FE40 EF    01    88  0BB0    754  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                00C8    30  0BB7    755  BSBW APT_COMMAND  ;
                                        00  0BBA    756 45$: HALT    ; Other unexpected Machine Check Exc.
                             26    0F    DA  0BBB    757 50$: MTPR # XF,#MCESR  ; Clear MCESR
                             27    0F    DA  0BBE    758  MTPR # XF,#CAER  ; Clear CAER
                             5E    8E    C0  0BC1    759  ADDL2 (SP)+,SP  ; Reset Stack Pointer
                        6E   FE3F CF    D0  0BC4    760  MOVL RET_PC,(SP)  ; Modify PC for return to start of
                                            0BC9    761      ; failing test
                                        02  0BC9    762 60$: REI    ; Return from exception
```

```
                     0BCA    763  .SBTTL Write Non_Existent Mem Routine
                     0BCA    764 ;**
                     0BCA    765 ; Write Non-Existent Memory Service Routine:
                     0BCA    766 ;
                     0BCA    767 ; This routine is set up to handle unexpected and expected
                     0BCA    768 ; write references to non-existent memory. At the start of
                     0BCA    769 ; each test, a NXM_WT_FLAG will be set or cleared dependent
                     0BCA    770 ; upon an expected or unexpected NXM Write reference. The
                     0BCA    771 ; service routine will first check if the flag is set. If
                     0BCA    772 ; set, the MCESR, CAER, and NXM_WT_FLAG wil be cleared and
                     0BCA    773 ; service routine will return to the test flow. Upon return,
                     0BCA    774 ; the flag will be tested again to determine if the expected
                     0BCA    775 ; interrupt did in fact, occur. In the case where a NXM Write
                     0BCA    776 ; reference was not expected, a valid error halt will occur with
                     0BCA    777 ; provisions to loop on that error by replacing the error halt
                     0BCA    778 ; with a NOP instruction.
                     0BCA    779 ;
                     0BCA    780 ; NOTE:  The purpose of the conditionals at the beginning of
                     0BCA    781 ;        the service routine is to protect Index locations
                     0BCA    782 ;        under test during the Cache Tag and Data Store
                     0BCA    783 ;        Dual Address Tests (Tests 6 & 7).
                     0BCA    784 ;
                     0BCA    785 ;-
                     0BCA    786  .ALIGN LONG
          00000BCC   0BCC    787  TEMP = . & ^XFFF  ; Put Index value into TEMP
          00000AC8   0BCC    788  .IF LE <TEMP- X104>  ; If Index is LE 104, make it 104
                     0BCC    789  . = . + < X104-TEMP>  ; Make Index equal to 104
                     0BCC    790  .IFF    ; If false, go to next conditional
          000009C8   0BCC    791  .IF LE <TEMP- X204>  ; If Index is LE 204, make it 204
                     0BCC    792  . = . + < X204-TEMP>  ; Make Index equal to 204
                     0BCC    793  .IFF    ; If false, go to next conditional
          000007C8   0BCC    794  .IF LE <TEMP- X404>  ; If Index is LE 404, make it 404
                     0BCC    795  . = . + < X404-TEMP>  ; Make Index equal to 404
                     0BCC    796  .IFF    ; If false, go to next conditional
          0000042C   0BCC    797  .IF LT <TEMP- X7A0>  ; If Index is LT 7A0, do not change
                     0BCC    798  .IFF    ; If false, go to next conditional
          000003C8   0BCC    799  .IF LE <TEMP- X804>  ; If Index is LE 804, make it 804
                     0BCC    800  . = . + < X804-TEMP>  ; Make Index equal to 804
                     0BCC    801  .IFF    ; If false, go to next conditional
          FFFFFC2C   0BCC    802  .IF GT<TEMP- XFA0>  ; If Index is GT FA0, move up to 104
                     0BCC    803  . = <. & ^XFFFFF000> + ^X1104 ; Move Index up to the next 104
                     0BCC    804  .ENDC    ; End conditional
                     0BCC    805  .ENDC    ; End conditional
                     0BCC    806  .ENDC    ; End conditional
                     0BCC    807  .ENDC    ; End conditional
                     0BCC    808  .ENDC    ; End conditional
                     0BCC    809  .ENDC    ; End conditional
          00000BCC   0BCC    810  BGNTST = .
                     0BCC    811 WT_NXM_SRV:    ; Continued on next line
  18 FE33 CF   00  E1 0BCC    812  BBC #0,NXM_WT_FLAG,10$ ; Check status of Wt. NXM Flag
          26   0F  DA 0BD2    813  MTPR #^XF,#MCESR  ; Clear MCESR
          27   0F  DA 0BD5    814  MTPR #^XF,#CAER  ; Clear CAER
          FE28 CF  94 0BD8    815  CLRB NXM_WT_FLAG  ; Clear flag
 0000FE50 EF   0D  B1 0BDC    816  CMPW #13,APT_TEST_NUM ; If Test 13, raise IPL to 1F
               2E  12 0BE3    817  BNEQ 20$   ; REI if not Test 13
          12   1F  DA 0BE5    818  MTPR #^X1F,#IPL  ; Raise IPL to 1F
               29  11 0BE8    819  BRB 20$   ; Return and continue test
```

```
                25    01    DA  0BEA    820 10$: MTPR #1,#CADR  ; Disable Cache
          12 0000FE00 EF    1F  E1  0BED  821  BBC #31,APT_CTRL_FLAGS,15$ ; Branch if not APT mode
          0000FE44 EF   07 AF   3E  0BF5  822  MOVAW B 15$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01   88  0BFD  823  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                    007B    30  0C04  824  BSBW APT_COMMAND  ;
                            00  0C07  825 15$: HALT    ; Unexpected NXM Write Error
                26    0F    DA  0C08  826  MTPR # XF,#MCESR  ; Clear MCESR
                27    0F    DA  0C0B  827  MTPR # XF,#CAER  ; Clear CAER
             6E   FDF5 CF    D0  0C0E  828  MOVL RET_PC,(SP)  ; Modify PC for return to start of
                                 0C13  829      ; failing test
                            02  0C13  830 20$: REI    ; Return from Interrupt
                        00000C14  0C14  831  ENDTST = .
                        FFFFFFF8  0C14  832  .IF GE<ENDTST-BGNTST- X50>  ; Check test length
                                  0C14  833  .ERROR ; MAXIMUM TEST LENGTH OF 50 HEX BYTES EXCEEDED
                                  0C14  834  .ENDC
                                  0C14  835 ;
```

```
                                0C14   836  .SBTTL Access Control Violation Rtn
                                0C14   837 ;**
                                0C14   838 ; Access Control Violation Service Routine:
                                0C14   839 ;
                                0C14   840 ; This routine is set up to determine expected and unexpected
                                0C14   841 ; Access Control Violations. If an expected ACV occurs, the
                                0C14   842 ; ACC_VIO_FLAG is cleared, the Stack Pointer is reset and
                                0C14   843 ; loaded with the PC Address following the instruction that
                                0C14   844 ; caused the ACV. This allows the REI instruction to cause
                                0C14   845 ; the program to continue the test flow.
                                0C14   846 ;
                                0C14   847 ;  An unexpected ACV will cause a HALT to occur. The Stack
                                0C14   848 ; will contain an error code and the initial VA which caused
                                0C14   849 ; the fault, as well as the PC and PSL.
                                0C14   850 ;
                                0C14   851 ; Looping on an ACV is possible by replacing the HALT instr.
                                0C14   852 ; with a NOP instruction.
                                0C14   853 ;
                                0C14   854 ;
                                0C14   855 ;--
                                0C14   856  .ALIGN LONG
                                0C14   857 ACV_SRV:     ; Continued on next line
        0E FDED CF    00    E1  0C14   858  BBC #0,ACC_VIO_FLAG,10$ ; Check for unexpected ACV error
                 FDE8 CF    94  0C1A   859  CLRB ACC_VIO_FLAG  ; Clear flag for expected ACV
                 5E   08    C0  0C1E   860  ADDL2 #8,SP    ; Reset the Stack Pointer
           6F    FDE6 CF    D0  0C21   861  MOVL CONT_PC,(SP)  ; Load Addr. to continue test
                      23    11  0C26   862  BRB 20$    ; Go to REI
   12 0000FE00 EF    1F    E1  0C28   863 10$: BBC #31,APT_CTRL_FLAGS,15$ ; Branch if not APT mode
   0000FE44 EF    42 AF    3E  0C30   864  MOVAW B 15$,APT_ERR_PC ; Save Error PC
      0000FE40 EF    01    88  0C38   865  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                   0040    30  0C3F   866  BSBW APT_COMMAND  ;
                           00  0C42   867 15$: HALT     ; ERROR, unexpected ACV.
                               0C43   868      ; Read Stack for VA ref.
                               0C43   869      ; that caused fault and
                               0C43   870      ; error code.
                 5E   08    C0  0C43   871  ADDL2 #8,SP    ; Reset  the Stack Pointer
           6E    FDBD CF    D0  0C46   872  MOVL RET_PC,(SP)  ; Modify Pc for return to start of
                               0C4B   873      ; failing test.
                           02  0C4B   874 20$: REI     ; Return from exception
```

```
                                0C4C    875  .SBTTL End Pass Routine
                                0C4C    876 ;++
                                0C4C    877 ; Endpass routine:
                                0C4C    878 ;
                                0C4C    879 ; This routine counts 10 passes of the Test Program,
                                0C4C    880 ; prints an end of pass message, and resumes the Test
                                0C4C    881 ; Program.
                                0C4C    882 ;
                                0C4C    883 ; NOTE: The number of passes between end-of-pass
                                0C4C    884 ;       typeout can be changed by deposited the
                                0C4C    885 ;       the desired number in address 230 before
                                0C4C    886 ;       starting the test.
                                0C4C    887 ;--
                                0C4C    888
                                0C4C    889 END_PASS::    ; Continued on next line
          0000FE54 EF   D6      0C4C    890  INCL APT_PASS_CNT  ; Bump the pass count once
    51  50 0000FE54 EF   0A   7B  0C52    891  EDIV #10,APT_PASS_CNT,R0,R1 ; Done 10 passes?
      15 0000FE00 EF   1F   E1  0C5B    892  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT
      07 0000FE00 EF   1B   E0  0C63    893  BBS #27,APT_CTRL_FLAGS,20$ ; Branch if inhibit typeout
                        51   D5  0C6B    894  TSTL R1   ; Done 10 passes?
                        03   12  0C6D    895  BNEQ 20$   ; Branch if no
                      00A7   30  0C6F    896  BSBW TYPE_END_PASS  ; Type end of pass message
                      000D   30  0C72    897 20$: BSBW APT_COMMAND  ; Check for APT monitor
                      470C   31  0C75    898  BRW PCS_INIT  ; Go to the PCS Init routine
                        51   D5  0C78    899 10$: TSTL R1   ; Done 10 passes?
                        03   12  0C7A    900  BNEQ 30$   ; Branch if no
                      009A   30  0C7C    901  BSBW TYPE_END_PASS
                      4702   31  0C7F    902 30$: BRW PCS_INIT  ; Go to the PCS Init routine
                             0C82    903  .SBTTL APT Command Routine
                             0C82    904 ;++
                             0C82    905 ; This routine is called when an error is detected or at the end of
                             0C82    906 ; a virtual pass. If the routine was called by the end pass routine
                             0C82    907 ; (the error flag is not set), the routine ignores any restart address
                             0C82    908 ; that APT may have said to perform since the end pass routine can only
                             0C82    909 ; restart the program.
                             0C82    910 ;--
                             0C82    911 APT_COMMAND::
    38 0000FE00 EF   1F   E1  0C82    912  BBC #31,APT_CTRL_FLAGS,20$ ; BRANCH IF NOT APT
       0000FE40 EF   01   B1  0C8A    913  CMPW #1,APT_MSG_TYPE  ; MESSAGE TYPE 1?
                        16   13  0C91    914  BEQL 10$   ; BRANCH IF YES
                     50   20   DB  0C93    915  MFPR #RXCS,R0  ; GET RECEIVER STATUS
                  28 50   07   E1  0C96    916  BBC #7,R0,20$  ; EXIT IF NO CHARACTER
                     50   21   DB  0C9A    917  MFPR #RXDB,R0  ; GET THE CHARACTER
                  50   80 8F   8A  0C9D    918  BICB #^X80,R0  ; CLEAR PARITY BIT
                     50   03   91  0CA1    919  CMPB #3,R0   ; CONTROL C?
                          1C   12  0CA4    920  BNEQ 20$   ; BRANCH IF NO
                        0093   30  0CA6    921  BSBW TYPE_PROMPT  ; TYPE THE PROMPT
0000FE44 EF   8000 8F   AA  0CA9    922 10$: BICW #^X8000,APT_ERR_PC ; MAKE SURE ERROR PC IS 15 BITS
                        000E   30  0CB2    923  BSBW CHECK_APT_CMD  ; CHECK FOR AN APT COMMAND
                0000FE40 EF   B4  0CB5    924  CLRW APT_MSG_TYPE  ; CLEAR THE MESSAGE TYPE
                          50   D5  0CBB    925  TSTL R0   ; RESTART ADDRESS SPECIFIED?
                          03   13  0CBD    926  BEQL 20$   ; BRANCH IF NO
                     6E   50   D0  0CBF    927  MOVL R0,(SP)   ; RETURN TO RESTART ADDRESS
                          05   0CC2    928 20$: RSB    ; RETURN
```

```
                                0CC3   929  .SBTTL Check APT Command Routine
                                0CC3   930 ;++
                                0CC3   931 ; This routine is called at the beginning of the program (if it was started
                                0CC3   932 ; at 200(x)) and by the APT_COMMAND routine. It performs the functions
                                0CC3   933 ; defined by the APT_CMD_WORD. If the command was type 3 or 5, it returns the
                                0CC3   934 ; start address of the program in R0 otherwise R0 is cleared.
                                0CC3   935 ;--
                                0CC3   936 CHECK_APT_CMD:
            50    20    DB      0CC3   937 20$: MFPR #RXCS,R0  ; CHECK FOR CTRL C CHARACTER
          0F 50    07    E1      0CC6   938  BBC #7,R0,5$  ; BRANCH IF NO CHARACTER
            50    21    DB      0CCA   939  MFPR #RXDB,R0  ; GET THE CHARACTER
          50   80 8F    8A      0CCD   940  BICB # X80,R0  ; CLEAR PARITY BIT
            50    03    91      0CD1   941  CMPB #3,R0   ; CTRL C?
                  03    12      0CD4   942  BNEQ 5$   ; BRANCH IF NO
                0063    30      0CD6   943  BSBW TYPE_PROMPT  ; TYPE THE PROMPT
     50   0000FE04'EF    B0      0CD9   944 5$: MOVW APT_CMD_WORD,R0  ; GET THE APT COMMAND
            50    03    91      0CE0   945  CMPB #3,R0   ; COMMAND 3?
                  0D    12      0CE3   946  BNEQ 12$   ; BRANCH IF NO
          0000FE00'EF    D4      0CE5   947  CLRL APT_CTRL_FLAGS  ; CLEAR THE CONTROL FLAGS
        50    FDC3 CF    DE      0CEB   948 11$: MOVAL START,R0  ; RETURN THE STARTING PC
                  20    11      0CF0   949  BRB EXIT   ; START TESTING
            50    05    91      0CF2   950 12$: CMPB #5,R0   ; COMMAND 5?
                  F4    13      0CF5   951  BEQL 11$   ; BRANCH IF YES
            50    06    91      0CF7   952  CMPB #6,R0   ; COMMAND 6?
                  0F    12      0CFA   953  BNEQ 14$   ; BRANCH IF NO
     0000FE40 EF    06    D0      0CFC   954  MOVL #6,APT_MSG_TYPE  ; SET THE MESSAGE TYPE
          0000FE04 EF    B4      0D03   955  CLRW APT_CMD_WORD  ; CLEAR ABORT COMMAND
                  B8    11      0D09   956  BRB 20$   ; HANG HERE
            50    09    91      0D0B   957 14$: CMPB #9,R0   ; COMMAND 9?
                  B3    12      0D0E   958  BNEQ 20$   ; BRANCH IF NO
                  50    D4      0D10   959  CLRL R0   ; RETURN WITH NO TRANSFER ADDRESS
     0000FE04 EF    B4      0D12   960 EXIT: CLRW APT_CMD_WORD
                  05    0D18   961  RSB    ; EXIT
```

```
                     0D19   962  .SBTTL Type End Pass Message Routine
                     0D19   963 ;++
                     0D19   964 ; This routine is used to type the End of Pass message.
                     0D19   965 ;--
                     0D19   966 TYPE_END_PASS:
        51    01     D0   0D19   967  MOVL #1,R1    ; Initialize index
  53    F52C CF     9A   0D1C   968  MOVZBL ENDPASS,R3  ; Get number of chars to print
              52     D4   0D21   969  CLRL R2    ; Clear character stuffer
52    F524 CF41     9A   0D23   970 5$: MOVZBL ENDPASS R1 ,R2  ; Get a character
            0005     30   0D29   971  BSBW PRINT_CHARACTER  ; TRANSMITT THE CHARACTER
   F3 51    53     F3   0D2C   972  AOBLEQ R3,R1,5$  ; Print the whole message
                   05   0D30   973  RSB     ; RETURN
```

```
                                0D31   974  .SBTTL Print Character Routine
                                0D31   975 ;++
                                0D31   976 ; This routine is used to transmit a character ( in R2) to the console
                                0D31   977 ; terminal port.
                                0D31   978 ;--
                                0D31   979 PRINT_CHARACTER:
    50    22    DB              0D31   980 10$: MFPR #TXCS,R0  ; Get tx status
F9  50    07    E1              0D34   981  BBC #7,R0,10$  ; Transmitter ready?
    23    52    DA              0D38   982  MTPR R2,#TXDB  ; Print a character
                05              0D3B   983  RSB    ; EXIT
```

```
                                0D3C    984  .SBTTL Print DS> Prompt to APT Routine
                                0D3C    985 ;++
                                0D3C    986 ; This routine is used to type the "DS> " promt to APT.
                                0D3C    987 ;--
                                0D3C    988 TYPE_PROMPT:
                     0F   BB    0D3C    989  PUSHR #^M<R0,R1,R2,R3> ; SAVE THE REGISTERS
                  51 01   D0    0D3E    990  MOVL #1,R1    ; INIT INDEX
               53 F522 CF   9A    0D41    991  MOVZBL PROMPT,R3  ; GET CHARACTER COUNT
            52 F51C CF41   9A    0D46    992 5$: MOVZBL PROMPT R1 ,R2  ; GET CHARACTER
                  FFE2   30    0D4C    993  BSBW PRINT_CHARACTER  ; PRINT IT
              F3 51 53   F3    0D4F    994  AOBLEQ R3,R1,5$  ; PRINT THEM ALL
                     0F   BA    0D53    995  POPR #^M<R0,R1,R2,R3> ; RESTORE THE REGISTERS
                          05    0D55    996  RSB    ; EXIT
```

```
                                0D56   997  .SBTTL Type 'No PCS' to user Routine
                                0D56   998 ;++
                                0D56   999 ; This routine is used to type the 'No PCS' message if not APT.
                                0D56  1000 ;--
                                0D56  1001 TYPE_NO_PCS:
                     0F   BB    0D56  1002  PUSHR # M<R0,R1,R2,R3> ; SAVE THE REGISTERS
                51   01   D0    0D58  1003  MOVL #1,R1    ; INIT INDEX
             53   F50D CF   9A  0D5B  1004  MOVZBL NO_PCS,R3  ; GET CHARACTER COUNT
          52   F507 CF41   9A  0D60  1005 5$: MOVZBL NO_PCS R1 ,R2  ; GET CHARACTER
                    FFC8   30  0D66  1006  BSBW PRINT_CHARACTER  ; PRINT IT
              F3 51   53   F3  0D69  1007  AOBLEQ R3,R1,5$  ; PRINT THEM ALL
                     0F   BA  0D6D  1008  POPR # M<R0,R1,R2,R3> ; RESTORE THE REGISTERS
                          05  0D6F  1009  RSB    ; EXIT
```

```
0D70  1010  .SBTTL General Notes
0D70  1011 ; General Cache Test notes:
0D70  1012 ;
0D70  1013 ; The first four test routines will be implemented in straight line
0D70  1014 ; code to eliminate the possibility of getting cache hits while
0D70  1015 ; executing test code.  Hardware faults might cause the program
0D70  1016 ; to behave unpredictably if the actual test code was stored in
0D70  1017 ; cache.
0D70  1018 ;
0D70  1019 ; A cache flush (invalidation of all cache locations) is performed
0D70  1020 ; when enabling the cache, (clearing bit<0> of the Cache Disable
0D70  1021 ; Register(CADR).
0D70  1022 ;
0D70  1023 ; It is important that Cache Tag Store Index locations used in the tests
0D70  1024 ; do not get referenced by the test code. This could cause misleading
0D70  1025 ; results. Depending on the type of test, one of two methods will be
0D70  1026 ; employed. The first method uses a Longword Storage Directive at the
0D70  1027 ; beginning of the test. Bits<23:12> of the Physical Address will be
0D70  1028 ; cleared leaving only the Index value, PAD<11:02>. This test Index
0D70  1029 ; will be unique, as long as the test is less than 1000 bytes, and
0D70  1030 ; can therefore be used for any desired Tag Store Addresses.
0D70  1031 ; The second method makes use of Conditional Assembly Block Directives,
0D70  1032 ; in most cases to protect more than one Index location for a particular
0D7C  1033 ; test.
0D70  1034 ;
0D70  1035 ;--
```

```
                                0D70  1036  .SBTTL  TEST 01:  CACHE  CONTROL LOGIC TEST
                                0D70  1037 ;++
                                0D70  1038 ; FUNCTIONAL DESCRIPTION:
                                0D70  1039 ;
                                0D70  1040 ; The first subtest enables the cache and makes sure that the
                                0D70  1041 ; Cache Error Register (CAER) is clear.  Subtests 2-5 will
                                0D70  1042 ; attempt to perform simple read/write hits and misses, each
                                0D70  1043 ; time referencing Cache Index location zero (PAD<11:02>=0.
                                0D70  1044 ; Cache hardware faults at this stage of the diagnostic will
                                0D70  1045 ; be difficult to isolate to the function since sections of
                                0D70  1046 ; all of the Cache logic will be brought into use.
                                0D70  1047 ;
                                0D70  1048 ;--
                                0D70  1049
                                0D70  1050  SEQ  $TN    ; Macro Call for Test Sequence Check
                                0D70        TEST_1::    ; Continued on next line
          0000FE50 EF    01    B0  0D70         MOVW #1,APT_TEST_NUM ; Load the Test Number
               0000FE52 EF    B6  0D77         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50 EF    B1  0D7D         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        0D83          APT_TEST_COUNT  ; Check for the correct test number
                          20   13  0D88         BEQL T1_S1    ; Sequence correct, continue test
          0000FE00'EF    1F    E1  0D8A         BBC #31,APT_CTRL_FLAGS,-
                          17       0D91          T1_HLT    ; Branch if not APT mode
     0000FE44 EF    00000DA9'EF    9E  0D92         MOVAB T1_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01    B0  0D9D         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       FEDB   30  0DA4         BSBW APT_COMMAND  ; Call APT routine
                          01   11  0DA7         BRB T1_S1    ; Continue
                               0DA9        T1_HLT:
                          00   0DA9         HALT    ; ERROR!!! Program out of sequence
                               0DAA        T1_S1:
        FC57 CF    FD AF    DE  0DAA         MOVAL T1_S1,RET_PC  ; Store return PC
                               0DB0
                        0251   31  0DB0  1051  BRW 5$    ; Go to start of test
                               0DB3  1052
                               0DB3  1053 ;
                               0DB3  1054 ; The following Conditionals are used to prohibit the test code from
                               0DB3  1055 ; referencing Cache Index location zero.
                               0DB3  1056 ;
                          00000DB3   0DB3  1057  TEMP = . &  XFFF  ; Put present Index value into TEMP
                          00000DAF   0DB3  1058  .IF LE <TEMP-004>  ; If Index is LE 004, make it 004
                                     0DB3  1059  . = . + <004-TEMP>  ; Make Index equal to 004
                                     0DB3  1060  .IFF     ; If false, go to next conditional
                          00000043   0DB3  1061  .IF GT<TEMP- XD70>  ; If Index is GT D70, move up to 004
                          00001004   0DB3  1062  . = <. &  XFFFFF000> +  X1004 ; Move up Index to 004
                                     1004  1063  .ENDC     ; End conditional
                                     1004  1064  .ENDC     ; End conditional
                          00001004   1004  1065  BGNTST = .
                                     1004  1066 ;
                                     1004  1067 ; The first section of code checks for the presence of a Unibus
                                     1004  1068 ; Exerciser Terminator (UET) Module. The UET is necessary to
                                     1004  1069 ; generate scope Sync Points and is used in Cache Test 13.
                                     1004  1070
               F9FC CF    01    88  1004  1071 5$: BISB2 #1,NXM_RD_FLAG  ; Set flag in event of NXM Read
     F9F9 CF    00001018 EF    DE  1009  1072  MOVAL 10$,CONT_PC  ; Store Cont Addr in event of NXM Read
               00FFF460 EF    B5  1012  1073  TSTW UET_ADDR  ; Check for UET s presence
          1F F9E8 CF    00    E0  1018  1074 10$: BBS #0,NXM_RD_FLAG,30$ ; If no mach check, start test 1
       14 0000FE00 EF    1F    E1  101E  1075  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
```

```
                0000FE44 EF   3A AF    3E  1026  1076  MOVAW B 20$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF   01    88  102E  1077  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                            FC4A    30  1035  1078  BSBW APT_COMMAND  ;
                              03    11  1038  1079  BRB 30$   ;
                                    00  103A  1080 20$: HALT    ; ERROR, UET did not respond when
                                        103B  1081      ; reading UET Addr Reg (FFF460).
                              C7    11  103B  1082  BRB 5$   ; Loop on error
                                        103D  1083
                                        103D  1084 ;
                                        103D  1085 ; Start first Cache test
                                        103D  1086 ;
                                        103D  1087
                      F9C3 CF   01    8A  103D  1088  30$: BICB2 #1,NXM_RD_FLAG  ; Clear flag
                      F9BD CF   01    8A  1042  1089  BICB2 #1,NXM_WT_FLAG  ; Clear flag
                           25   00    DA  1047  1090  MTPR #0,#CADR  ; Enable Cache and perform Cache flush
                           51   25    DB  104A  1091  MFPR #CADR,R1  ; Read Cache Disable Register (CADR)
                           25   01    DA  104D  1092  MTPR #1,#CADR  ; Disable Cache
                         1F 51   00    E1  1050  1093  BBC #0,R1,50$  ; If Cache enabled, continue test
                14 0000FE00'EF   1F    E1  1054  1094  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
                0000FE44 EF   70 AF    3E  105C  1095  MOVAW B 40$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF   01    88  1064  1096  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                            FC14    30  106B  1097  BSBW APT_COMMAND
                              03    11  106E  1098  BRB 50$
                                    00  1070  1099 40$: HALT    ; ERROR, cannot enable Cache
                                        1071  1100      ; R1 contains contents of CADR
                              CA    11  1071  1101  BRB 30$   ; Loop on error
                           27   0F    DA  1073  1102 50$: MTPR # XF,#CAER  ; Clear Cache Error Register (CAER)
                           25   00    DA  1076  1103  MTPR #0,#CADR  ; Enable Cache
                           51   27    DB  1079  1104  MFPR #CAER,R1  ; Read CAER into R1
                           25   01    DA  107C  1105  MTPR #1,#CADR  ; Disable Cache
                           51   00    91  107F  1106  CMPB #0,R1   ; Check if the CAER is clear
                                1F    13  1082  1107  BEQL T1_S2   ; If clear, go to next subtest
                14 0000FE00'EF   1F    E1  1084  1108  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
                0000FE44 EF   A0'AF    3E  108C  1109  MOVAW B 60$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF   01    88  1094  1110  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                            FBE4    30  109B  1111  BSBW APT_COMMAND
                              03    11  109E  1112  BRB 70$
                                    00  10A0  1113 60$: HALT    ; ERROR, cannot clear the CAER
                                        10A1  1114      ; R1 contains the contents of CAER
                              D0    11  10A1  1115  BRB 50$   ; Loop on error
                                        10A3  1116 70$:
                                        10A3  1117 ;
                                        10A3  1118 ; Subtest 2 makes a write reference to PA zero after a Cache flush.
                                        10A3  1119 ; The reference should cause a Cache miss.
                                        10A3  1120 ;
                                        10A3  1121
                    F95E CF   FD AF    DE  10A3  1122 T1_S2: MOVAL T1_S2,RET_PC  ; Store return PC
                           27   0F    DA  10A9  1123  MTPR # XF,#CAER  ; Clear the CAER
                                50    D4  10AC  1124  CLRL R0   ; Load R0 with PA=0
                           25   00    DA  10AE  1125  MTPR #0,#CADR  ; Enable Cache
                        00FFF460 EF    B5  10B1  1126  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                                60    D4  10B7  1127  CLRL (R0)   ; Write zeroes into PA zero
                           51   27    DB  10B9  1128  MFPR #CAER,R1  ; Store hit/miss information
                           25   01    DA  10BC  1129  MTPR #1,#CADR  ; Disable Cache
                         1F 51   00    E1  10BF  1130  BBC #0,R1,T1_S3  ; If no miss occurs, continue test
                14 0000FE00 EF   1F    E1  10C3  1131  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
                0000FE44 EF   DF AF    3E  10CB  1132  MOVAW B 10$,APT_ERR_PC ; Save Error PC
```

```
                0000FE40'EF    01    88   10D3  1133  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          FBA5    30   10DA  1134  BSBW APT_COMMAND
                            03    11   10DD  1135  BRB 40$
                                  00   10DF  1136 10$: HALT     ; ERROR, unexpected Cache hit occured
                                       10E0  1137       ; R1 contains the contents of CAER
                            C1    11   10E0  1138  BRB T1_S2   ; Loop on error
                                       10E2  1139 40$:
                                       10E2  1140 ;
                                       10E2  1141 ; Subtest 3 first writes zeroes into PA zero, then makes a read
                                       10E2  1142 ; reference to PA zero which should result in a Read hit.  The
                                       10E2  1143 ; data read from the Cache Data Store is checked for zero.
                                       10E2  1144 ;
                                       10E2  1145
                 F91F CF    FD AF    DE   10E2  1146 T1_S3: MOVAL T1_S3,RET_PC  ; Store return PC
                       27    0F    DA   10E8  1147  MTPR #^XF,#CAER  ; Clear the CAER
                             50    D4   10EB  1148  CLRL R0    ; Load PA=0 into R0
                       25    00    DA   10ED  1149  MTPR #0,#CADR  ; Flush the Cache
                             60    D4   10F0  1150 5$: CLRL (R0)    ; Write zeroes into PA zero
                       52    60    D0   10F2  1151  MOVL (R0),R2   ; Read PA zero, and store into R2
                       51    27    DB   10F5  1152  MFPR #CAER,R1  ; Store hit/miss information
                       25    01    DA   10F9  1153  MTPR #1,#CADR  ; Disable Cache
                    1F 51    00    E0   10FB  1154  BBS #0,R1,20$   ; If read hit, continue test
                14 0000FE00 EF    1F    E1   10FF  1155  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
                0000FE44 EF    1B'AF    3E   1107  1156  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF    01    88   110F  1157  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                             FB69    30   1116  1158  BSBW APT_COMMAND
                               03    11   1119  1159  BRB 20$
                                     00   111B  1160 10$: HALT     ; ERROR, did not get Cache hit
                                          111C  1161       ; R1 contains contents of CAER
                               D2    11   111C  1162  BRB 5$    ; Loop on error
                          52    00    D1   111E  1163 20$: CMPL #0,R2    ; Check data from Cache
                                1F    13   1121  1164  BEQL T1_S4   ; If data correct, go to next subtest
                14 0000FE00 EF    1F    E1   1123  1165  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
                0000FE44'EF    3F'AF    3E   112B  1166  MOVAW B 30$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF    01    88   1133  1167  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                             FB45    30   113A  1168  BSBW APT_COMMAND
                               03    11   113D  1169  BRB 40$
                                     00   113F  1170 30$: HALT     ; ERROR, data read incorrect
                                          1140  1171       ; should be zero.
                                          1140  1172       ; R2 contains data read
                               AE    11   1140  1173  BRB 5$    ; Loop on error
                                          1142  1174 40$:
                                          1142  1175 ;
                                          1142  1176 ; Subtest 4 writes zeroes into PA 03F000  and checks that a miss occurs.
                                          1142  1177 ; The uniqueness of PA 03F000 is that it allows as  many bits to be set
                                          1142  1178 ; in the Cache Tag Store without causing a Non-Existent Memory reference
                                          1142  1179 ; to be made. The minimum memory is 256 kilobytes. References to NXM
                                          1142  1180 ; are checked in the next test.
                                          1142  1181 ;
                                          1142  1182
                  F8BF CF    FD AF    DE   1142  1183 T1_S4: MOVAL T1_S4,RET_PC  ; Store return PC
                        27    0F    DA   1148  1184  MTPR #^XF,#CAER  ; Clear the CAER
                50    0003F000 8F    D0   114B  1185  MOVL #^X3F000,R0  ; Load PA to be written into R0
                        25    00    DA   1152  1186  MTPR #0,#CADR  ; Flush the Cache
                     00FFF460 EF    B5   1155  1187  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                              60    D4   115B  1188  CLRL (R0)    ; Write zeroes into PA 03F000
                        51    27    DB   115D  1189  MFPR #CAER,R1  ; Store hit/miss information
```

```
                25   01    DA  1160  1190  MTPR #1,#CADR  ; Disable Cache
             1F 51   00    E1  1163  1191  BBC #0,R1,T1_S5  ; If no hit, continue test
       14 0000FE00 EF   1F   E1  1167  1192  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
       0000FE44'EF   83'AF   3E  116F  1193  MOVAW B 10$,APT_ERR_PC ; Save Error PC
          0000FE40 EF   01   88  1177  1194  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                   FB01   30  117E  1195  BSBW APT_COMMAND
                     03   11  1181  1196  BRB 40$
                          00  1183  1197 10$: HALT    ; ERROR, unexpected hit
                              1184  1198      ; referencing PA 03F000
                              1184  1199      ; R1 contains contents of CAER
                     BC   11  1184  1200  BRB T1_S4   ; Loop on error
                              1186  1201 40$:
                              1186  1202 ;
                              1186  1203 ; Subtest 5 writes ones into PA 03F000, reads and checks for a read hit,
                              1186  1204 ; and verifies that the correct data is read.
                              1186  1205 ;
                              1186  1206
         F87B CF    FD AF    DE  1186  1207 T1_S5: MOVAL T1_S5,RET_PC  ; Store return PC
                27   0F    DA  118C  1208  MTPR # XF,#CAER  ; Clear the CAER
          50    0003F000 8F   D0  118F  1209  MOVL # X3F000,R0  ; Load PA into R0
                25   00    DA  1196  1210  MTPR #0,#CADR  ; Flush the Cache
          60    FFFFFFFF 8F   D0  1199  1211 5$: MOVL # XFFFFFFFF,(R0) ; Write all ones into PA 03F000
                52   60    D0  11A0  1212  MOVL (R0),R2   ; Read PA 03F000, store data into R2
                51   27    DB  11A3  1213  MFPR #CAER,R1  ; Store hit/miss information
                25   01    DA  11A6  1214  MTPR #1,#CADR  ; Disable Cache
             1F 51   00    E0  11A9  1215  BBS #0,R1,20$  ; If read hit, continue test
       14 0000FE00 EF   1F   E1  11AD  1216  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
       0000FE44 EF   C9 AF   3E  11B5  1217  MOVAW B 10$,APT_ERR_PC ; Save Error PC
          0000FE40 EF   01   88  11BD  1218  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                   FABB   30  11C4  1219  BSBW APT_COMMAND
                     03   11  11C7  1220  BRB 20$
                          00  11C9  1221 10$: HALT    ; ERROR, did not get hit reading
                              11CA  1222      ; PA 03F000
                     CD   11  11CA  1223  BRB 5$   ; Loop on error
          52    FFFFFFFF 8F   D1  11CC  1224 20$: CMPL # XFFFFFFFF,R2  ; Check data read from Cache
                     1F   13  11D3  1225  BEQL TEST_2   ; If data correct, go to next test
       14 0000FE00 EF   1F   E1  11D5  1226  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
       0000FE44 EF   F1 AF   3E  11DD  1227  MOVAW B 30$,APT_ERR_PC ; Save Error PC
          0000FE40 EF   01   88  11E5  1228  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                   FA93   30  11EC  1229  BSBW APT_COMMAND
                     03   11  11EF  1230  BRB 40$
                          00  11F1  1231 30$: HALT    ; ERROR, data incorrect
                              11F2  1232      ; data written = FFFFFFFF
                              11F2  1233      ; R2 contains data read
                     A5   11  11F2  1234  BRB 5$   ; Loop on error
                              11F4  1235 40$:
                    000011F4  11F4  1236  ENDTST = .
                    FFFFFFA0  11F4  1237  .IF GE<ENDTST-BGNTST- X250> ; Check test length
                              11F4  1238  .ERROR ; MAXIMUM TEST LENGTH OF 250 HEX BYTES EXCEEDED!
                              11F4  1239  .ENDC    ; End conditional
```

```
                                11F4  1241  .SBTTL  TEST 02:  NON-EXISTENT MEMORY REFERENCE TEST
                                11F4  1242
                                11F4  1243 ;**
                                11F4  1244 ; FUNCTIONAL DESCRIPTION:
                                11F4  1245 ;
                                11F4  1246 ; Since the minumum amount of memory required is 256 kilobytes,
                                11F4  1247 ; reference to Physical Address (PAD)<23:18> may cause a Non-
                                11F4  1248 ; Existent Write Reference Interrupt to occur. The referenced
                                11F4  1249 ; Cache location however, will be cached. This test will verify
                                11F4  1250 ; this operation by attempting to write into a location with the Tag
                                11F4  1251 ; Store equal to EFF which will be non-existent memory. If an
                                11F4  1252 ; interrupt is expected, as it will be in this case, a Write NXM
                                11F4  1253 ; Interrupt will occur and the program will go to a service routine
                                11F4  1254 ; and simply return from the interrupt and continue executing the main
                                11F4  1255 ; code. This non-existent memory reference will be necessary to test
                                11F4  1256 ; all the Tag and Data Store locations in  the Dynamic Tests.
                                11F4  1257 ; This test also verifies that all bits in the Tag Store can be
                                11F4  1258 ; set.
                                11F4  1259 ;
                                11F4  1260 ;--
                                11F4  1261
                                11F4  1262
                                11F4  1263  SEQ  $TN    ; Macro Call for Test Sequence check
                                11F4        TEST_2::    ; Continued on next line
        0000FE50 EF   02   B0   11F4         MOVW #2,APT_TEST_NUM ; Load the Test Number
             0000FE52 EF   B6   11FB         INCW APT_TEST_COUNT  ; Bump up the test count
             0000FE50 EF   B1   1201         CMPW APT_TEST_NUM,-  ; Continued on next line
             0000FE52 EF        1207          APT_TEST_COUNT  ; Check for the correct test number
                      20   13   120C         BEQL T2_S1    ; Sequence correct, continue test
             0000FE00 EF   1F   E1   120E    BBC #31,APT_CTRL_FLAGS,-
                           17        1215     T2_HLT    ; Branch if not APT mode
0000FE44 EF   0000122D EF   9E   1216         MOVAB T2_HLT,APT_ERR_PC ; Save the error PC
             0000FE40 EF   01   B0   1221     MOVW #1,APT_ERR_FLAG  ; Set the error flag
                        FA57   30   1228     BSBW APT_COMMAND  ; Call APT routine
                           01   11   122B     BRB T2_S1    ; Continue
                                122D        T2_HLT:
                                00   122D     HALT    ; ERROR!!! Program out of sequence
                                122E        T2_S1:
         F7D3 CF    FD AF   DE   122E       MOVAL T2_S1,RET_PC  ; Store return PC
                                1234
                           06   11   1234  1264  BRB STRT2   ; Go to start of test
                                1236  1265  .ALIGN LONG   ; Index value must be long aligned
                     00000000   1238  1266 INDEX1:  .LONG 0   ; Reserve location for Cache Index
               50    F9 AF   DE   123C  1267 STRT2: MOVAL INDEX1,R0  ; Store reserved location
         50    00FFF000 8F   CA   1240  1268  BICL2 # XFFF000,R0  ; Retain Index value
                  25   00   DA   1247  1269  MTPR #0,#CADR  ; Flush the Cache
                  27   0F   DA   124A  1270  MTPR # XF,#CAER  ; Clear the CAER
              F7B2 CF   01   88   124D  1271  BISB2 #1,NXM_HT_FLAG  ; Set flag for expected interrupt
         50    00EFF000 8F   C8   1252  1272  BISL2 # XEFF000,R0  ; Load a NXM address into R0
               00FFF460 EF   B5   1259  1273  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
         60    AAAAAAAA 8F   D0   125F  1274  MOVL # XAAAAAAAA,(R0) ; Write A s to NXM location (Tag=EFF)
                        01   1266  1275  NOP    ; Wait for interrupt to occur
          20 F798 CF    00   E1   1267  1276  BBC #0,NXM_HT_FLAG,20$ ; If interrupt occured, continue test
       14 0000FE00 EF    1F   E1   126D  1277  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
       0000FE44 EF    89 AF   3E   1275  1278  MOVAW B 10$,APT_ERR_PC ; Save Error PC
          0000FE40 EF    01   88   127D  1279  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       F9FB   30   1284  1280  BSBW APT_COMMAND
```

```
                    04   11  1287  1281  BRB 20$
                         00  1289  1282 10$: HALT     ; ERROR, interrupt did not occur
                             128A  1283       ; when writing to NXM location
                  FFA1   31  128A  1284  BRW T2_S1    ; Loop on error
              52    60   D0  128D  1285 20$: MOVL (R0),R2    ; Read address spec by R0 (Tag=EFF)
              51    27   DB  1290  1286  MFPR #CAER,R1   ; Store hit/miss information
           20 51    00   E0  1293  1287  BBS #0,R1,40$   ; If Cache hit, continue test
14 0000FE00'EF     1F   E1  1297  1288  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
0000FE44 EF   B3'AF     3E  129F  1289  MOVAW B 30$,APT_ERR_PC ; Save Error PC
   0000FE40'EF     01   88  12A7  1290  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                  F9D1   30  12AE  1291  BSBW APT_COMMAND
                    04   11  12B1  1292  BRB 40$
                         00  12B3  1293 30$: HALT     ; Did not get Cache hit when
                             12B4  1294       ; reading address spec by R0
                  FF77   31  12B4  1295  BRW T2_S1    ; Loop on error
   52   AAAAAAAA 8F     D1  12B7  1296 40$: CMPL #^XAAAAAAAA,R2  ; Check data read from Cache
                    20   13  12BE  1297  BEQL TEST_3    ; If data correct, continue test
14 0000FE00 EF     1F   E1  12C0  1298  BBC #31,APT_CTRL_FLAGS,50$ ; Branch if not APT mode
0000FE44 EF   DC AF     3E  12C8  1299  MOVAW B 50$,APT_ERR_PC ; Save Error PC
   0000FE40 EF     01   88  12D0  1300  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                  F9A8   30  12D7  1301  BSBW APT_COMMAND
                    04   11  12DA  1302  BRB 60$
                         00  12DC  1303 50$: HALT     ; ERROR, data incorrect
                             12DD  1304       ; Data written = AAAAAAAA
                             12DD  1305       ; R2 contains data read
                  FF4E   31  12DD  1306  BRW T2_S1    ; Loop on error
                             12E0  1307 60$:
```

```
                        12E0  1308  .SBTTL  TEST 03:  SIMPLE ADDRESS PATHS TEST
                        12E0  1309
                        12E0  1310 ;**
                        12E0  1311 ; FUNCTIONAL DESCRIPTION:
                        12E0  1312 ;
                        12E0  1313 ; This test checks the input and output paths of the cache Tag
                        12E0  1314 ; Store rams for bits S-A-0, S-A-1, and shorted together by
                        12E0  1315 ; writing a minimal bit fault pattern set to the Tag Store
                        12E0  1316 ; Rams and checking for hits.
                        12E0  1317 ;
                        12E0  1318 ; Four patterns are used in this test, they are as follows:
                        12E0  1319 ;
                        12E0  1320 ; Tag Store Test Pattern1 =  AAA
                        12E0  1321 ; Tag Store Test Pattern2 =  555
                        12E0  1322 ; Tag Store Test Pattern3 =  333
                        12E0  1323 ; Tag Store Test Pattern4 =  E0F
                        12E0  1324 ;
                        12E0  1325 ; The Parity Generator, Checker, and the Hit/Miss Comparitors
                        12E0  1326 ; (equals checker) are implicitly checked in this test.
                        12E0  1327 ;
                        12E0  1328 ; The test is subdivided into four subtests, one for each pattern
                        12E0  1329 ; used. Each subtest will make a write reference to the respective
                        12E0  1330 ; test pattern address. Since all four patterns will address non-
                        12E0  1331 ; existent memory, a Write NXM Interrupt will occur. The NXM_WT_FLAG
                        12E0  1332 ; is tested after the write reference to insure that the expected
                        12E0  1333 ; interrupt occured. A read reference is then performed to that
                        12E0  1334 ; test address and it is checked for a read hit.
                        12E0  1335 ;
                        12E0  1336 ; Failures in this test would manifest themselves by either halting
                        12E0  1337 ; at the respective subtest or causing an unexpected Read NXM Machine
                        12E0  1338 ; Check to occur. The Read NXM Machine Check would be the result of
                        12E0  1339 ; writing the wrong test address.
                        12E0  1340 ;
                        12E0  1341 ;--
                        12E0  1342
                        12E0  1343
                        12E0  1344  SEQ  $TN    ; Macro for test sequence check
                        12E0       TEST_3::    ; Continued on next line
        0000FE50 EF    03    B0   12E0         MOVW #3,APT_TEST_NUM ; Load the Test Number
             0000FE52 EF    B6   12E7         INCW APT_TEST_COUNT  ; Bump up the test count
             0000FE50 EF    B1   12ED         CMPW APT_TEST_NUM,-  ; Continued on next line
             0000FE52 EF         12F3          APT_TEST_COUNT  ; Check for the correct test number
                        20  13   12F8         BEQL T3_S1    ; Sequence correct, continue test
        0000FE00 EF    1F    E1   12FA         BBC #31,APT_CTRL_FLAGS,-
                        17        1301          T3_HLT    ; Branch if not APT mode
0000FE44 EF    00001319 EF    9E   1302         MOVAB T3_HLT,APT_ERR_PC ; Save the error PC
        0000FE40 EF    01    B0   130D         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                     F96B    3C   1314         BSBW APT_COMMAND  ; Call APT routine
                        01    11   1317         BRB T3_S1    ; Continue
                             1319        T3_HLT:
                        00   1319         HALT     ; ERROR!!! Program out of sequence
                             131A        T3_S1:
         F6E7 CF    FD AF    DE   131A         MOVAL T3_S1,RET_PC  ; Store return PC
                             1320
                             1320  1345
                             1320  1346 ;
                             1320  1347 ; Subtest 1 writes a Tag Store = AAA, checks that an expected interrupt
```

```
                                1320  1348 ; occured, then reads that Tag Store, this time checking for a hit.
                                1320  1349 ; The particular Index value used for this test is specified by
                                1320  1350 ; the Index value of label 'INDEX2'.
                                1320  1351 ;
                                1320  1352
                                1320  1353
                          06    11  1320  1354  BRB STRT3    ; Go to start of test
                                    1322  1355  .ALIGN LONG    ; Index value must be long aligned
                    00000000    1324  1356 INDEX2:  .LONG 0    ; Reserve test location
              50    F9 AF   DE  1328  1357 STRT3: MOVAL INDEX2,R0   ; Store location
        50    00FFF000 8F   CA  132C  1358  BICL2 # XFFF000,R0   ; Retain Index value
              25    00   DA  1333  1359  MTPR #0,#CADR   ; Flush the Cache
              27    0F   DA  1336  1360  MTPR # XF,#CAER   ; Clear the CAER
        50    00AAA000 8F   C8  1339  1361  BISL2 # XAAA000,R0   ; Load Test address into R0
           F6BF CF    01   88  1340  1362  BISB2 #1,NXM_WT_FLAG   ; Set flag for expected write to NXM
            00FFF460 EF   B5  1345  1363  TSTW SYNC_POINT   ; Provide Sync on Unibus Master Sync
                          60   D4  134B  1364  CLRL (R0)    ; Write Tag Store = AAA
                          01   88  134D  1365  NOP     ; Wait for expected Interrupt to occur
     1F F6B1 CF    00   E1  134E  1366  BBC #0,NXM_WT_FLAG,20$ ; If interrupt occured, continue test
14 0000FE00'EF    1F   E1  1354  1367  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
0000FE44'EF    70'AF   3E  135C  1368  MOVAW B 10$,APT_ERR_PC ; Save Error PC
   0000FE40 EF    01   88  1364  1369  BISB2 #1,APT_ERR_FLAG   ; Set Error Flag
                 F914   30  136B  1370  BSBW APT_COMMAND
                    03   11  136E  1371  BRB 20$
                          00   1370  1372 10$: HALT     ; ERROR, Write NXM Interrupt did not
                                    1371  1373       ; occur when ref. Tag Store AAA
                          A7   11  1371  1374  BRB T3_S1    ; Loop on error
                          60   D5  1373  1375 20$: TSTL (R0)    ; Read Tag Store AAA
                51    27   DB  1375  1376  MFPR #CAER,R1   ; Store hit/miss information
             20 51    00   E0  1378  1377  BBS #0,R1,T3_S2   ; If hit, continue test
14 0000FE00 EF    1F   E1  137C  1378  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
0000FE44'EF    98 AF   3E  1384  1379  MOVAW B 30$,APT_ERR_PC ; Save Error PC
   0000FE40 EF    01   88  138C  1380  BISB2 #1,APT_ERR_FLAG   ; Set Error Flag
                 F8EC   30  1393  1381  BSBW APT_COMMAND
                    04   11  1396  1382  BRB 40$
                          00   1398  1383 30$: HALT     ; Did not get Cache hit when
                                    1399  1384       ; reading Tag Store AAA
                 FF7E   31  1399  1385  BRW T3_S1    ; Loop on error
                                139C  1386 40$:
                                139C  1387 ;
                                139C  1388 ; Subtest 2 writes a Tag Store = 555, checks that ar expected interrupt
                                139C  1389 ; occured, then reads that Tag Store, this time checking for a hit.
                                139C  1390 ; The particular Index value used for this test is specified by
                                139C  1391 ; the Index value of label 'INDEX2'.
                                139C  1392 ;
                                139C  1393
     F665 CF     FD AF   DE  139C  1394 T3_S2: MOVAL T3_S2,RET_PC   ; Store return PC
              25    00   DA  13A2  1395  MTPR #0,#CADR   ; Flush the Cache
              27    0F   DA  13A5  1396  MTPR # XF,#CAER   ; Clear the CAER
        50    FF78 CF   DE  13A8  1397  MOVAL INDEX2,R0   ; Store location
  50    00FFF000 8F   CA  13AD  1398  BICL2 # XFFF000,R0   ; Retain Index value
  50    00D55000 8F   C8  13B4  1399  BISL2 # XD55000,R0   ; Load Test address into R0
        F644 CF    01   88  13BB  1400  BISB2 #1,NXM_WT_FLAG   ; Set flag for expected write to NXM
         00FFF460 EF   B5  13C0  1401  TSTW SYNC_POINT   ; Provide Sync on Unibus Master Sync
                          60   D4  13C6  1402  CLRL (R0)    ; Write Tag Store = 555
                          01   88  13C8  1403  NOP     ; Wait for expected Interrupt to occur
     1F F636 CF    00   E1  13C9  1404  BBC #0,NXM_WT_FLAG,20$ ; If interrupt occured, continue test
```

```
                                                  1405  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
                                                  1406  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                                                  1407  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                                  1408  BSBW APT_COMMAND
                                                  1409  BRB 20$
                                                  1410 10$: HALT    ; ERROR, Write NXM Interrupt did not
                                                  1411       ; occur when ref. Tag Store 555
                                                  1412  BRB T3_S2   ; Loop on error
                                                  1413 20$: TSTL (R0)   ; Read Tag Store 555
                                                  1414  MFPR #CAER,R1  ; Store hit/miss information
                                                  1415  BBS #0,R1,T3_S3  ; If hit, continue test
                                                  1416  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
                                                  1417  MOVAW B 30$,APT_ERR_PC ; Save Error PC
                                                  1407  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                                  140E  BSBW APT_COMMAND
                                                  1411  BRB 40$
                                                  1413  1421 30$: HALT    ; Did not get Cache hit when
                                                  1414  1422       ; reading Tag Store 555
                                                  1414  1423  BRB T3_S2   ; Loop on error
                                                  1416  1424 40$:
                                                  1416  1425 ;
                                                  1416  1426 ; Subtest 3 writes a Tag Store = 333, checks that an expected interrupt
                                                  1416  1427 ; occured, then reads that Tag Store, this time checking for a hit.
                                                  1416  1428 ; The particular Index value used for this test is specified by
                                                  1416  1429 ; the Index value of label 'INDEX2'.
                                                  1416  1430 ;
                                                  1416  1431
                                                  1416  1432 T3_S3: MOVAL T3_S3,RET_PC  ; Store return PC
                                                  141C  1433  MTPR #0,#CADR  ; Flush the Cache
                                                  141F  1434  MTPR # XF,#CAER  ; Clear the CAER
                                                  1422  1435  MOVAL INDEX2,R0  ; Store location
                                                  1427  1436  BICL2 # XFFF000,R0  ; Retain Index value
                                                  142E  1437  BISL2 # XB33000,R0  ; Load Test address into R0
                                                  1435  1438  BISB2 #1,NXM_WT_FLAG  ; Set flag for expected write to NXM
                                                  143A  1439  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                                                  1440  1440  CLRL (R0)   ; Write Tag Store = 333
                                                  1442  1441  NOP     ; Wait for expected Interrupt to occur
                                                  1443  1442  BBC #0,NXM_WT_FLAG,20$ ; If interrupt occured, continue test
                                                  1449  1443  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
                                                  1451  1444  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                                                  1459  1445  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                                  1460  1446  BSBW APT_COMMAND
                                                  1463  1447  BRB 20$
                                                  1465  1448 10$: HALT    ; ERROR, Write NXM Interrupt did not
                                                  1466  1449       ; occur when ref. Tag Store = 333
                                                  1466  1450  BRB T3_S3   ; Loop on error
                                                  1468  1451 20$: TSTL (R0)   ; Read Tag Store = 333
                                                  146A  1452  MFPR #CAER,R1  ; Store hit/miss information
                                                  146D  1453  BBS #0,R1,T3_S4  ; If hit, continue test
                                                  1471  1454  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
                                                  1479  1455  MOVAW B 30$,APT_ERR_PC ; Save Error PC
                                                  1481  1456  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                                  1488  1457  BSBW APT_COMMAND
                                                  148B  1458  BRB 40$
                                                  148D  1459 30$: HALT    ; Did not get Cache hit when
                                                  148E  1460       ; reading Tag Store 333
                                                  148E  1461  BRB T3_S3   ; Loop on error
```

```
                        1490  1462 40$:
                        1490  1463 ;
                        1490  1464 ; Subtest 4 writes a Tag Store = F0F, checks that an expected interrupt
                        1490  1465 ; occured, then reads that Tag Store, this time checking for a hit.
                        1490  1466 ; The particular Index value used for this test is specified by
                        1490  1467 ; the Index value of label 'INDEX2'.
                        1490  1468 ;
                        1490  1469
     F571 CF    FD AF   DE   1490  1470 T3_S4: MOVAL T3_S4,RET_PC  ; Store return PC
            25    00   DA   1496  1471  MTPR #0,#CADR  ; Flush the Cache
            27    0F   DA   1499  1472  MTPR # XF,#CAER  ; Clear the CAER
        50    FE84 CF   DE   149C  1473  MOVAL INDEX2,R0  ; Store location
    50    00FFF000 8F   CA   14A1  1474  BICL2 # XFFF000,R0  ; Retain Index value
    50    00E0F000 8F   C8   14A8  1475  BISL2 # XE0F000,R0  ; Load Test address into R0
        F550 CF    01   88   14AF  1476  BISB2 #1,NXM_WT_FLAG  ; Set flag for expected write to NXM
         00FFF460'EF   B5   14B4  1477  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                60    D4   14BA  1478  CLRL (R0)   ; Write Tag Store = E0F
                      01   14BC  1479  NOP    ; Wait for expected Interrupt to occur
      1F F542 CF    00   E1   14BD  1480  BBC #0,NXM_WT_FLAG,20$ ; If interrupt occured, continue test
  14 0000FE00'EF    1F   E1   14C3  1481  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
 0000FE44'EF    DF'AF   3E   14CB  1482  MOVAW B 10$,APT_ERR_PC ; Save Error PC
    0000FE40'EF    01   88   14D3  1483  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
              F7A5   30   14DA  1484  BSBW APT_COMMAND
                03   11   14DD  1485  BRB 20$
                     00   14DF  1486 10$: HALT    ; ERROR, Write NXM Interrupt did not
                          14E0  1487       ; when ref. Tag Store = E0F
                AE   11   14E0  1488  BRB T3_S4   ; Loop on error
                60   D5   14E2  1489 20$: TSTL (R0)   ; Read Tag Store = E0F
          51    27   DB   14E4  1490  MFPR #CAER,R1  ; Store hit/miss information
       1F 51    00   E0   14E7  1491  BBS #0,R1,TEST_4  ; If hit, continue test
  14 0000FE00 EF    1F   E1   14EB  1492  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
 0000FE44'EF    07'AF   3E   14F3  1493  MOVAW B 30$,APT_ERR_PC ; Save Error PC
    0000FE40 EF    01   88   14FB  1494  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
              F77D   30   1502  1495  BSBW APT_COMMAND
                03   11   1505  1496  BRB 40$
                     00   1507  1497 30$: HALT    ; Did not get Cache hit when
                          1508  1498       ; reading Tag Store = E0F
                86   11   1508  1499  BRB T3_S4   ; Loop on error
                          150A  1500 40$:
```

```
                     150A  1501  .SBTTL TEST 04:  SIMPLE DATA PATHS TEST
                     150A  1502
                     150A  1503 ;++
                     150A  1504 ; FUNCTIONAL DESCRIPTION:
                     150A  1505 ;
                     150A  1506 ; This test checks the input and output paths of the Cache Data
                     150A  1507 ; Store Rams and Output Driver for bits S-A-0, S-A-1, and
                     150A  1508 ; shorted together by writing a minimal bit fault pattern set
                     150A  1509 ; across Cache<31:00> while checking for correct data.
                     150A  1510 ;
                     150A  1511 ; The test is divided into six subtests each of which writes a
                     150A  1512 ; test pattern into the Data Store, reads the Data Store and checks
                     150A  1513 ; for the correct data results.
                     150A  1514 ;
                     150A  1515 ; The test location used is in the 10000 range with an Index value
                     150A  1516 ; from the label 'Index3'.
                     150A  1517 ;
                     150A  1518 ; The six test patterns required are as follows:
                     150A  1519 ;
                     150A  1520 ; Data Test Pattern1 = AAAAAAAA
                     150A  1521 ; Data Test Pattern2 = 55555555
                     150A  1522 ; Data Test Pattern3 = 33333333
                     150A  1523 ; Data Test Pattern4 = 0F0F0F0F
                     150A  1524 ; Data Test Pattern5 = 00FF00FF
                     150A  1525 ; Data Test Pattern6 = 0000FFFF
                     150A  1526 ;
                     150A  1527 ; The logic implicitly tested here includes parity generation,
                     150A  1528 ; checking and the Index line drivers.
                     150A  1529 ;
                     150A  1530 ;--
                     150A  1531
                     150A  1532  SEQ  $TN    ; Macro call for Test Sequence check
                     150A        TEST_4::    ; Continued on next line
          0000FE50'EF    04   B0  150A         MOVW #4,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  1511         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  1517         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        151D          APT_TEST_COUNT  ; Check for the correct test number
                          20   13  1522         BEQL T4_S1    ; Sequence correct, continue test
          0000FE00'EF    1F   E1  1524         BBC #31,APT_CTRL_FLAGS,-
                          17      152B          T4_HLT    ; Branch if not APT mode
0000FE44 EF    00001543'EF   9E  152C         MOVAB T4_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01   B0  1537         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                        F741   30  153E         BSBW APT_COMMAND  ; Call APT routine
                          01   11  1541         BRB T4_S1    ; Continue
                               00  1543        T4_HLT:
                                   1543         HALT    ; ERROR!!! Program out of sequence
                                   1544        T4_S1:
           F4BD CF    FD AF   DE  1544         MOVAL T4_S1,RET_PC  ; Store return PC
                                   154A
                          04   11  154A  1533  BRB STRT4    ; Go to start of test
                                   154C  1534  .ALIGN LONG    ; Index value must be long aligned
                    00000000       154C  1535 INDEX3:  .LONG 0    ; Reserve location for Index value
                50    F9 AF   DE  1550  1536 STRT4: MOVAL INDEX3,R0  ; Store location
          50    00FFF000 8F   CA  1554  1537  BICL2 # XFFF000,R0  ; Retain Index value
                    25   00   DA  155B  1538  MTPR #0,#CADR  ; Flush the Cache
                    27   0F   DA  155E  1539  MTPR # XF,#CAER  ; Clear the CAER
          50    00010000 8F   C8  1561  1540  BISL2 # X10000,R0  ; Load PA into R0
```

```
                51   55555555 8F    D0  1568  1541  MOVL #X55555555,R1  ; Load Test pattern into R1
                        60   51    D0  156F  1542 10$: MOVL R1,(R0)    ; Write Test pattern 1 into Cache
                        52   60    D0  1572  1543  MOVL (R0),R2    ; Read Test pattern and store in R2
                    53  52   51    CD  1575  1544  XORL3 R1,R2,R3  ; Check data results
                             1F    13  1579  1545  BEQL T4_S2    ; If data correct, go to next subtest
             14 0000FE00'EF  1F    E1  157B  1546  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
             0000FE44'EF   97'AF   3E  1583  1547  MOVAW B 20$,APT_ERR_PC ; Save Error PC
                0000FE40'EF  01    88  158B  1548  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          F6ED    30  1592  1549  BSBW APT_COMMAND
                             03    11  1595  1550  BRB 40$
                             00    11  1597  1551 20$: HALT     ; ERROR, data incorrect
                                       1598  1552       ; R0 contains referenced location
                                       1598  1553       ; R1 contains data written
                                       1598  1554       ; R2 contains data read
                                       1598  1555       ; R3 contains bad data bit positions
                             D5    11  1598  1556  BRB 10$    ; Loop on error
                                       159A  1557 40$:
                 F467 CF     FD AF    DE  159A  1558 T4_S2: MOVAL T4_S2,RET_PC  ; Store return PC
                        25   00    DA  15A0  1559  MTPR #0,#CADR  ; Flush the Cache
                        27   0F    DA  15A3  1560  MTPR #XF,#CAER  ; Clear the CAER
                     50   A3  AF    DE  15A6  1561  MOVAL INDEX3,R0  ; Store location
                50   00FFF000 8F    CA  15AA  1562  BICL2 #XFFF000,R0  ; Retain Index value
                50   00010000 8F    C8  15B1  1563  BISL2 #X10000,R0  ; Load PA into R0
                51   AAAAAAAA 8F    D0  15B8  1564  MOVL #XAAAAAAAA,R1  ; Load Test pattern into R1
                        60   51    D0  15BF  1565 10$: MOVL R1,(R0)    ; Write Test pattern 2
                        52   60    D0  15C2  1566  MOVL (R0),R2    ; Read Test pattern 2 and store in R2
                    53  52   51    CD  15C5  1567  XORL3 R1,R2,R3  ; Check data results
                             1F    13  15C9  1568  BEQL T4_S3    ; If data correct, go to next subtest
             14 0000FE00 EF  1F    E1  15CB  1569  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
             0000FE44 EF   E7'AF   3E  15D3  1570  MOVAW B 20$,APT_ERR_PC ; Save Error PC
                0000FE40 EF  01    88  15DB  1571  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          F69D    30  15E2  1572  BSBW APT_COMMAND
                             03    11  15E5  1573  BRB 40$
                             00    11  15E7  1574 20$: HALT     ; ERROR, data incorrect
                                       15E8  1575       ; R0 contains referenced location
                                       15E8  1576       ; R1 contains data written
                                       15E8  1577       ; R2 contains data read
                                       15E8  1578       ; R3 contains bad data bit positions
                             D5    11  15E8  1579  BRB 10$    ; Loop on error
                                       15EA  1580 40$:
                 F417 CF     FD AF    DE  15EA  1581 T4_S3: MOVAL T4_S3,RET_PC  ; Store return PC
                        25   00    DA  15F0  1582  MTPR #0,#CADR  ; Flush the Cache
                        27   0F    DA  15F3  1583  MTPR #XF,#CAER  ; Clear the CAER
                    50   FF52  CF    DE  15F6  1584  MOVAL INDEX3,R0  ; Store location
                50   00FFF000 8F    CA  15FB  1585  BICL2 #XFFF000,R0  ; Retain Index value
                50   00010000 8F    C8  1602  1586  BISL2 #X10000,R0  ; Load PA into R0
                51   33333333 8F    D0  1609  1587  MOVL #X33333333,R1  ; Load Test pattern into R1
                        60   51    D0  1610  1588 10$: MOVL R1,(R0)    ; Write Test pattern 3
                        52   60    D0  1613  1589  MOVL (R0),R2    ; Read Test pattern 3 and store in R2
                    53  52   51    CD  1616  1590  XORL3 R1,R2,R3  ; Check data results
                             1F    13  161A  1591  BEQL T4_S4    ; If data correct, go to next subtest
             14 0000FE00 EF  1F    E1  161C  1592  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
             0000FE44'EF   38 AF   3E  1624  1593  MOVAW B 20$,APT_ERR_PC ; Save Error PC
                0000FE40 EF  01    88  162C  1594  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          F64C    30  1633  1595  BSBW APT_COMMAND
                             03    11  1636  1596  BRB 40$
                             00    11  1638  1597 20$: HALT     ; ERROR, data incorrect
```

```
                                  1639  1598      ; R0 contains referenced location
                                  1639  1599      ; R1 contains data written
                                  1639  1600      ; R2 contains data read
                                  1639  1601      ; R3 contains bad data bit positions
                        D5   11   1639  1602  BRB 10$   ; Loop on error
                                  163B  1603 40$:
         F3C6 CF    FD AF    DE   163B  1604 T4_S4: MOVAL T4_S4,RET_PC  ; Store return PC
                 25    00    DA   1641  1605  MTPR #0,#CADR  ; Flush the Cache
                 27    0F    DA   1644  1606  MTPR # XF,#CAER  ; Clear the CAER
              50    FF01 CF    DE   1647  1607  MOVAL INDEX3,R0  ; Store location
           50    00FFF000 8F    CA   164C  1608  BICL2 # XFFF000,R0  ; Retain Index value
           50    00010000 8F    C8   1653  1609  BISL2 # X10000,R0  ; Load PA into R0
           51    0F0F0F0F 8F    D0   165A  1610  MOVL # X0F0F0F0F,R1  ; Load Test pattern into R1
                 60    51    D0   1661  1611 10$: MOVL R1,(R0)   ; Write Test pattern 4
                 52    60    D0   1664  1612  MOVL (R0),R2   ; Read Test pattern 4 and store in R2
              53    52    51    CD   1667  1613  XORL3 R1,R2,R3  ; Check data results
                        1F    13   166B  1614  BEQL T4_S5   ; If data correct, go to next subtest
      14 0000FE00'EF    1F    E1   166D  1615  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
      0000FE44 EF    89'AF    3E   1675  1616  MOVAW B 20$,APT_ERR_PC ; Save Error PC
         0000FE40'EF    01    88   167D  1617  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                     F5FB    30   1684  1618  BSBW APT_COMMAND
                        03    11   1687  1619  BRB 40$
                              00   1689  1620 20$: HALT    ; ERROR, data incorrect
                                  168A  1621      ; R0 contains referenced location
                                  168A  1622      ; R1 contains data written
                                  168A  1623      ; R2 contains data read
                                  168A  1624      ; R3 contains bad data bit positions
                        D5   11   168A  1625  BRB 10$   ; Loop on error
                                  168C  1626 40$:
         F375 CF    FD AF    DE   168C  1627 T4_S5: MOVAL T4_S5,RET_PC  ; Store return PC
                 25    00    DA   1692  1628  MTPR #0,#CADR  ; Flush the Cache
                 27    0F    DA   1695  1629  MTPR # XF,#CAER  ; Clear the CAER
              50    FEB0 CF    DE   1698  1630  MOVAL INDEX3,R0  ; Store location
           50    00FFF000 8F    CA   169D  1631  BICL2 # XFFF000,R0  ; Retain Index value
           50    00010000 8F    C8   16A4  1632  BISL2 # X10000,R0  ; Load PA into R0
           51    00FF00FF 8F    D0   16AB  1633  MOVL # X00FF00FF,R1  ; Load Test pattern into R1
                 60    51    D0   16B2  1634 10$: MOVL R1,(R0)   ; Write Test pattern 5
                 52    60    D0   16B5  1635  MOVL (R0),R2   ; Read Test pattern 5 and store in R2
              53    52    51    CD   16B8  1636  XORL3 R1,R2,R3  ; Check data results
                        1F    13   16BC  1637  BEQL T4_S6   ; If data correct, go to next subtest
      14 0000FE00 EF    1F    E1   16BE  1638  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
      0000FE44 EF    DA AF    3E   16C6  1639  MOVAW B 20$,APT_ERR_PC ; Save Error PC
         0000FE40'EF    01    88   16CE  1640  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                     F5AA    30   16D5  1641  BSBW APT_COMMAND
                        03    11   16D8  1642  BRB 40$
                              00   16DA  1643 20$: HALT    ; ERROR, data incorrect
                                  16DB  1644      ; R0 contains referenced location
                                  16DB  1645      ; R1 contains data written
                                  16DB  1646      ; R2 contains data read
                                  16DB  1647      ; R3 contains bad data bit positions
                        D5   11   16DB  1648  BRB 10$   ; Loop on error
                                  16DD  1649 40$:
         F324 CF    FD AF    DE   16DD  1650 T4_S6: MOVAL T4_S6,RET_PC  ; Store return PC
                 25    00    DA   16E3  1651  MTPR #0,#CADR  ; Flush the Cache
                 27    0F    DA   16E6  1652  MTPR # XF,#CAER  ; Clear the CAER
              50    FE5F CF    DE   16E9  1653  MOVAL INDEX3,R0  ; Store location
           50    00FFF000 8F    CA   16EE  1654  BICL2 # XFFF000,R0  ; Retain Index value
```

```
                50    00010000 8F    C8  16F5  1655  BISL2 # X10000,R0  ; Load PA into R0
                51    0000FFFF 8F    D0  16FC  1656  MOVL # X0000FFFF,R1  ; Load Test pattern into R1
                          60   51    D0  1703  1657 10$: MOVL R1,(R0)   ; Write Test pattern 6
                          52   60    D0  1706  1658  MOVL (R0),R2   ; Read Test pattern 6 and store in R2
                     53   52   51    CD  1709  1659  XORL3 R1,R2,R3  ; Check data results
                                1F    13  170D  1660  BEQL TEST_5   ; If data correct, go to next test
             14 0000FE00'EF     1F    E1  170F  1661  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
             0000FE44'EF   2B'AF    3E  1717  1662  MOVAW B 20$,APT_ERR_PC ; Save Error PC
                0000FE40 EF    01    88  171F  1663  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                             F559   30  1726  1664  BSBW APT_COMMAND
                                03    11  1729  1665  BRB 40$
                                      00  172B  1666 20$: HALT    ; ERROR, data incorrect
                                          172C  1667       ; R0 contains referenced location
                                          172C  1668       ; R1 contains data written
                                          172C  1669       ; R2 contains data read
                                          172C  1670       ; R3 contains bad data bit positions
                                D5    11  172C  1671  BRB 10$   ; Loop on error
                                          172E  1672 40$:
```

```
                                172E  1673  .SBTTL TEST 05:  HIT/MISS COMPARITORS TEST
                                172E  1674
                                172E  1675 ;**
                                172E  1676 ; FUNCTIONAL DESCRIPTION:
                                172E  1677 ;
                                172E  1678 ; This test will verify the input paths to the Hit/Miss Comparitors
                                172E  1679 ; for bits S-A-0, S-A-1, shorted together, and logic functionality
                                172E  1680 ; by referencing Index location zero and alternately floating a one
                                172E  1681 ; across each side of the Hit/Miss comparitors and checking for
                                172E  1682 ; misses.
                                172E  1683 ;
                                172E  1684 ; Floating a one across one side of the Hit/miss comparitors
                                172E  1685 ; generates the following test patterns:
                                172E  1686 ;
                                172E  1687 ; pattern 1 = 1000
                                172E  1688 ; pattern 2 = 2000
                                172E  1689 ; pattern 3 = 4000
                                172E  1690 ; pattern 4 = 8000
                                172E  1691 ; pattern 5 =      10000
                                172E  1692 ; pattern 6 =      20000
                                172E  1693 ; pattern 7 =      40000
                                172E  1694 ; pattern 8 =      80000
                                172E  1695 ; pattern 9 =     100000
                                172E  1696 ; pattern 10 =     200000
                                172E  1697 ; pattern 11 =     400000
                                172E  1698 ; pattern 12 =     800000
                                172E  1699 ;
                                172E  1700 ;--
                                172E  1701
                                172E  1702  SEQ  $TN    ; Macro call for Test sequence check
                                172E       TEST_5::     ; Continued on next line
          0000FE50'EF    05    B0  172E          MOVW #5,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  1735          INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  173B          CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        1741           APT_TEST_COUNT  ; Check for the correct test number
                          20   13  1746          BEQL T5_S1    ; Sequence correct, continue test
          0000FE00'EF    1F    E1  1748          BBC #31,APT_CTRL_FLAGS,-
                          17       174F           T5_HLT    ; Branch if not APT mode
0000FE44'EF    00001767'EF    9E  1750          MOVAB T5_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01    B0  175B          MOVW #1,APT_ERR_FLAG  ; Set the error flag
                        F51D   30  1762          BSBW APT_COMMAND  ; Call APT routine
                          01   11  1765          BRB T5_S1    ; Continue
                               1767         T5_HLT:
                          00   1767          HALT     ; ERROR!!! Program out of sequence
                               1768         T5_S1:
           F299 CF    FD AF    DE  1768          MOVAL T5_S1,RET_PC  ; Store return PC
                               176E
                        0000   31  176E  1703  BRW 5$    ; Go to start of test
                               1771  1704
                               1771  1705 ; The following Conditional Assembly Block Directives are to
                               1771  1706 ; protect Cache Index location zero, PAD<11:02>=0, from being
                               1771  1707 ; referenced by the test code.
                               1771  1708
                           00000771  1771  1709  TEMP = . & ^XFFF  ; Put present Index value into TEMP
                           0000076D  1771  1710  .IF LE <TEMP-004>  ; If Index is LE 004, make it 004
                               1771  1711  . = . + <004-TEMP>  ; Make Index equal to 004
                               1771  1712  .IFF     ; If false, go to next conditional
```

```
                    FFFFFAC1  1771  1713  .IF GT <TEMP- XCB0>  ; If Index is GT CB0, move up to 004
                              1771  1714  . = <. & ^XFFFFF000> + ^X1004 ; Move up Index to 004
                              1771  1715  .ENDC    ; End conditional
                              1771  1716  .ENDC    ; End conditional
                    00001771  1771  1717  BGNTST = .
                              1771  1718 ;
                              1771  1719 ; Subtest 1 starts by making Test Address 1 (which is initially 1000)
                              1771  1720 ; a hit in the Cache Tag Store.  A read reference is then made to
                              1771  1721 ; PA zero which should result in a Cache miss. A new Test Address
                              1771  1722 ; is generated each time by shifting left one place, and making it
                              1771  1723 ; a hit. PA zero is read again, and this should result in a miss. This
                              1771  1724 ; process is continued for Cache Tag<23:12> inputs to the Hit/Miss
                              1771  1725 ; Comparitors.
                              1771  1726 ;
                              1771  1727 ; Patterns 1-6 are loaded into Cache via read references.
                              1771  1728 ;
                              1771  1729
            25    00    DA    1771  1730 5$: MTPR #0,#CADR  ; Flush the Cache
            27    0F    DA    1774  1731  MTPR #^XF,#CAER  ; Clear the CAER
      50    00001000  8F    D0    1777  1732  MOVL #^X1000,R0  ; Load Test Address 1 into R0
                  51    D4    177E  1733  CLRL R1    ; Load Test Address 2 into R1
            54    01    D0    1780  1734  MOVL #1,R4   ; Initialize counter
            25    00    DA    1783  1735 10$: MTPR #0,#CADR  ; Flush the Cache
      00FFF460'EF    BS    1786  1736  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                  60    D5    178C  1737  TSTL (R0)    ; Read Test Address 1
            52    27    DB    178E  1738  MFPR #CAER,R2  ; Store hit/miss information
         20 52    00    E1    1791  1739  BBC #0,R2,30$  ; If Cache miss, continue test
14 0000FE00 EF    1F    E1    1795  1740  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
0000FE44 EF    B1 AF    3E    179D  1741  MOVAW B 20$,APT_ERR_PC ; Save Error PC
   0000FE40 EF    01    88    17A5  1742  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
               F4D3    30    17AC  1743  BSBW APT_COMMAND
                  04    11    17AF  1744  BRB 30$
                        00    17B1  1745 20$: HALT    ; ERROR, unexpected Cache hit
                              17B2  1746      ; when reading Test Address 1
                              17B2  1747      ; R0 contains Test Address 1
               FFCE    31    17B2  1748  BRW 10$    ; Loop on error
                  60    D5    17B5  1749 30$: TSTL (R0)    ; Read Test Address 1 again
            52    27    DB    17B7  1750  MFPR #CAER,R2  ; Store hit/miss information
         1F 52    00    E0    17BA  1751  BBS #0,R2,50$  ; If Cache hit, continue test
14 0000FE00 EF    1F    E1    17BE  1752  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
0000FE44 EF    DA AF    3E    17C6  1753  MOVAW B 40$,APT_ERR_PC ; Save Error PC
   0000FE40 EF    01    88    17CE  1754  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
               F4AA    30    17D5  1755  BSBW APT_COMMAND
                  03    11    17D8  1756  BRB 50$
                        00    17DA  1757 40$: HALT    ; ERROR, did not get Cache hit
                              17DB  1758      ; when reading Test Address 1 again.
                              17DB  1759      ; R0 contains Test Address 1
                  D8    11    17DB  1760  BRB 30$    ; Loop on error
                  61    D5    17DD  1761 50$: TSTL (R1)    ; Read Test Address 2
            52    27    DB    17DF  1762  MFPR #CAER,R2  ; Store hit/miss information
         20 52    00    E1    17E2  1763  BBC #0,R2,70$  ; If Cache miss, continue test
14 0000FE00 EF    1F    E1    17E6  1764  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
0000FE44 EF    02 AF    3E    17EE  1765  MOVAW B 60$,APT_ERR_PC ; Save Error PC
   0000FE40 EF    01    88    17F6  1766  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
               F482    30    17FD  1767  BSBW APT_COMMAND
                  04    11    1800  1768  BRB 70$
                        00    1802  1769 60$: HALT    ; ERROR, unexpected Cache hit
```

```
                                    1803  1770       ; when reading Test Address 2
                                    1803  1771       ; R1 contains Test Address 2
                        FF7D   31   1803  1772  BRW 10$    ; Loop on error
                50   50   01   78   1806  1773 70$: ASHL #1,R0,R0  ; Generate new Test Address 1
                          54   D6   180A  1774  INCL R4
                     06   54   91   180C  1775  CMPB R4,#6
                          03   14   180F  1776  BGTR 75$
                        FF6F   31   1811  1777  BRW 10$
                                    1814  1778 75$:
                                    1814  1779
                                    1814  1780 ; Test patterns 7-12 are loaded into Cache via write references. This
                                    1814  1781 ; is necessary since the patterns may be NXM and therefore must be
                                    1814  1782 ; written to load into Cache.
                                    1814  1783
                     12   1F   DA   1814  1784  MTPR # X1F,#IPL  ; Raise IPL to block WT NXM Intr.
                     25   00   DA   1817  1785  MTPR #0,#CADR  ; Flush the Cache
           50   00040000 8F   D0   181A  1786  MOVL # X40000,R0  ; Load Test Address 1 into R0
                          51   D4   1821  1787  CLRL R1    ; Load Test Address 2 into R1
                     54   01   D0   1823  1788  MOVL #1,R4    ; Initialize counter
                     25   00   DA   1826  1789 80$: MTPR #0,#CADR  ; Flush the Cache
                00FFF460  EF   B5   1829  1790  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                          60   D4   182F  1791  CLRL (R0)    ; Reference Test Address 1
                          60   D5   1831  1792  TSTL (R0)    ; Read Test Address 1
                     52   27   DB   1833  1793  MFPR #CAER,R2  ; Store hit/miss information
                   1F 52   00   E0   1836  1794  BBS #0,R2,100$  ; If Cache hit, continue test
          14 0000FE00 EF   1F   E1   183A  1795  BBC #31,APT_CTRL_FLAGS,90$ ; Branch if not APT mode
          0000FE44 EF   56 AF   3E   1842  1796  MOVAW B 90$,APT_ERR_PC ; Save Error PC
             0000FE40 EF   01   88   184A  1797  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        F42E   30   1851  1798  BSBW APT_COMMAND
                          03   11   1854  1799  BRB 100$
                          00   1856  1800 90$: HALT     ; ERROR, did not get Cache hit
                                    1857  1801       ; when reading Test Address 1 again.
                                    1857  1802       ; R0 contains Test Address 1
                          CD   11   1857  1803  BRB 80$    ; Loop on error
                          61   D5   1859  1804 100$: TSTL (R1)    ; Read Test Address 2
                     52   27   DB   185B  1805  MFPR #CAER,R2  ; Store hit/miss information
                   1F 52   00   E1   185E  1806  BBC #0,R2,120$  ; If Cache miss, continue test
          14 0000FE00 EF   1F   E1   1862  1807  BBC #31,APT_CTRL_FLAGS,110$ ; Branch if not APT mode
          0000FE44 EF   7E AF   3E   186A  1808  MOVAW B 110$,APT_ERR_PC ; Save Error PC
             0000FE40 EF   01   88   1872  1809  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        F406   30   1879  1810  BSBW APT_COMMAND
                          03   11   187C  1811  BRB 120$
                          00   187E  1812 110$: HALT     ; ERROR, unexpected Cache hit
                                    187F  1813       ; when reading Test Address 2
                                    187F  1814       ; R1 contains Test Address 2
                          A5   11   187F  1815  BRB 80$    ; Loop on error
                50   50   01   78   1881  1816 120$: ASHL #1,R0,R0  ; Generate new Test Address 1
                  9D 54   06   F3   1885  1817  AOBLEQ #6,R4,80$  ; Repeat test for 6 test addresses
                F176 CF   01   88   1889  1818  BISB2 #1,NXM_WT_FLAG  ; Set flag for pending Wt. Intr.
                     12   00   DA   188E  1819  MTPR #0,#IPL    ; Allow interrupt to occur
                                    1891  1820
                                    1891  1821 ;
                                    1891  1822 ; Subtest 2 reverses the process of subtest 1 by floating a one
                                    1891  1823 ; across the PAD<23:12> inputs of the Hit/Miss Comparitors,
                                    1891  1824 ; reading a Cache Tag Store of zero each time, and checking for misses.
                                    1891  1825 ;
                                    1891  1826 ; Test Patterns 1-6 are tested by read references.
```

```
                                        1891  1827
          F170 CF    FD AF    DE      1891  1828 T5_S2: MOVAL T5_S2,RET_PC  ; Store return PC
                  25    00    DA      1897  1829  MTPR #0,#CADR  ; Flush the Cache
                  27    0F    DA      189A  1830  MTPR # XF,#CAER  ; Clear the CAER
                        50    D4      189D  1831  CLRL R0   ; Load Test Address 1 into R0
        51    00001000 8F    D0      189F  1832  MOVL # X1000,R1  ; Load Test Address 2 into R1
                  54    01    D0      18A6  1833  MOVL #1,R4   ; Initialize counter
                  25    00    DA      18A9  1834 10$: MTPR #0,#CADR  ; Flush the Cache
               00FFF460 EF    B5      18AC  1835  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                        60    D5      18B2  1836  TSTL (R0)   ; Read Test Address 1
                  52    27    DB      18B4  1837  MFPR #CAER,R2  ; Store hit/miss information
                1F 52    00    E1      18B7  1838  BBC #0,R2,30$  ; If Cache miss, continue test
          14 0000FE00 EF    1F    E1      18BB  1839  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
          0000FE44 EF    D7 AF    3E      18C3  1840  MOVAW B 20$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88      18CB  1841  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        F3AD    30      18D2  1842  BSBW APT_COMMAND
                          03    11      18D5  1843  BRB 30$
                                00      18D7  1844 20$: HALT    ; ERROR, unexpected Cache hit
                                        18D8  1845      ; when reading Test Address 1
                                        18D8  1846      ; R0 contains Test Address 1
                          CF    11      18D8  1847  BRB 10$   ; Loop on error
                          60    D5      18DA  1848 30$: TSTL (R0)   ; Read Test Address 1 again
                  52    27    DB      18DC  1849  MFPR #CAER,R2  ; Store hit/miss information
                1F 52    00    E0      18DF  1850  BBS #0,R2,50$  ; If Cache hit, continue test
          14 0000FE00 EF    1F    E1      18E3  1851  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
          0000FE44 EF    FF AF    3E      18EB  1852  MOVAW B 40$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88      18F3  1853  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        F385    30      18FA  1854  BSBW APT_COMMAND
                          03    11      18FD  1855  BRB 50$
                                00      18FF  1856 40$: HALT    ; ERROR, did not get Cache hit
                                        1900  1857      ; when reading Test Address 1 again.
                                        1900  1858      ; R0 contains Test Address 1
                          D8    11      1900  1859  BRB 30$   ; Loop on error
                          61    D5      1902  1860 50$: TSTL (R1)   ; Read Test Address 2
                  52    27    DB      1904  1861  MFPR #CAER,R2  ; Store hit/miss information
                20 52    00    E1      1907  1862  BBC #0,R2,70$  ; If Cache miss, continue test
          14 0000FE00 EF    1F    E1      190B  1863  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
          0000FE44 EF    27 AF    3E      1913  1864  MOVAW B 60$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88      191B  1865  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        F35D    30      1922  1866  BSBW APT_COMMAND
                          04    11      1925  1867  BRB 70$
                                00      1927  1868 60$: HALT    ; ERROR, unexpected Cache hit
                                        1928  1869      ; when reading Test Address 2
                                        1928  1870      ; R1 contains Test Address 2
                        FF7E    31      1928  1871  BRW 10$   ; Loop on error
                51    51    01    78      192B  1872 70$: ASHL #1,R1,R1  ; Generate new Test Address 2
                          54    D6      192F  1873  INCL R4
                  06    54    91      1931  1874  CMPB R4,#6
                          03    14      1934  1875  BGTR 75$
                        FF70    31      1936  1876  BRW 10$
                                        1939  1877 ; AOBLEQ #6,R4,10$  ; Repeat test for 6 test addresses
                                        1939  1878 75$:
                                        1939  1879
                                        1939  1880 ; Test patterns 7-12 are also tested by read references.
                                        1939  1881
                  25    00    DA      1939  1882  MTPR #0,#CADR  ; Flush the Cache
                          50    D4      193C  1883  CLRL R0   ; Load Test Address 1 into R0
```

```
                51    00040000 8F    D0  193E  1884  MOVL #X40000,R1  ; Load Test Address 2 into R1
                          54    01    D0  1945  1885  MOVL #1,R4   ; Initialize counter
                          25    00    DA  1948  1886 80$: MTPR #0,#CADR  ; Flush the Cache
                      00FFF460'EF    B5  194B  1887  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                                60    D5  1951  1888  TSTL (R0)   ; Read Test Address 1
                             52   27    DB  1953  1889  MFPR #CAER,R2  ; Store hit/miss information
                          1F 52   00    E1  1956  1890  BBC #0,R2,100$  ; If Cache miss, continue test
                   14 0000FE00 EF   1F    E1  195A  1891  BBC #31,APT_CTRL_FLAGS,90$ ; Branch if not APT mode
                   0000FE44'EF    76 AF    3E  1962  1892  MOVAW B 90$,APT_ERR_PC ; Save Error PC
                      0000FE40 EF   01    88  196A  1893  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                  F30E  30    1971  1894  BSBW APT_COMMAND
                                    03  11    1974  1895  BRB 100$
                                        00    1976  1896 90$: HALT    ; ERROR, unexpected Cache hit
                                              1977  1897       ; when reading Test Address 1
                                              1977  1898       ; R0 contains Test Address 1
                                    CF  11    1977  1899  BRB 80$   ; Loop on error
                                    60  D5    1979  1900 100$: TSTL (R0)   ; Read Test Address 1 again
                             52   27    DB  197B  1901  MFPR #CAER,R2  ; Store hit/miss information
                          1F 52   00    E0  197E  1902  BBS #0,R2,120$  ; If Cache hit, continue test
                   14 0000FE00'EF   1F    E1  1982  1903  BBC #31,APT_CTRL_FLAGS,110$ ; Branch if not APT mode
                   0000FE44'EF    9E'AF    3E  198A  1904  MOVAW B 110$,APT_ERR_PC ; Save Error PC
                      0000FE40'EF   01    88  1992  1905  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                  F2E6  30    1999  1906  BSBW APT_COMMAND
                                    03  11    199C  1907  BRB 120$
                                        00    199E  1908 110$: HALT    ; ERROR, did not get Cache hit
                                              199F  1909       ; when reading Test Address 1 again.
                                              199F  1910       ; R0 contains Test Address 1
                                    D8  11    199F  1911  BRB 100$   ; Loop on error
                          F05F CF   01    88  19A1  1912 120$: BISB2 #1,NXM_RD_FLAG  ; Set flag in event of NXM Reads
                F05C CF    000019B4'EF    DE  19A6  1913  MOVAL 125$,CONT_PC  ; Store Cont PC in event of NXM Read
                                61    D5  19AF  1914  TSTL (R1)   ; Read Test Address 2
                             52   27    DB  19B1  1915  MFPR #CAER,R2  ; Store hit/miss information
                       24 F04C CF   00    E1  19B4  1916 125$: BBC #0,NXM_RD_FLAG,140$ ; If Mach. Chk occurs, Cache miss has
                                              19BA  1917       ; occured, therefore continue test.
                          20 52   00    E1  19BA  1918  BBC #0,R2,140$  ; If no Mach. Chk. and Cache miss,
                                              19BE  1919       ; continue test.
                   14 0000FE00'EF   1F    E1  19BE  1920  BBC #31,APT_CTRL_FLAGS,130$ ; Branch if not APT mode
                   0000FE44 EF    DA'AF    3E  19C6  1921  MOVAW B 130$,APT_ERR_PC ; Save Error PC
                      0000FE40 EF   01    88  19CE  1922  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                  F2AA  30    19D5  1923  BSBW APT_COMMAND
                                    04  11    19D8  1924  BRB 140$
                                        00    19DA  1925 130$: HALT    ; ERROR, unexpected Cache hit
                                              19DB  1926       ; when reading Test Address 2
                                              19DB  1927       ; R1 contains Test Address 2
                                  FF6A  31    19DB  1928  BRW 80$   ; Loop on error
                          51    51    01    78  19DE  1929 140$: ASHL #1,R1,R1  ; Generate new Test Address 2
                                    54  D6    19E2  1930  INCL R4
                                06  54  91    19E4  1931  CMPB R4,#6
                                    03  14    19E7  1932  BGTR 145$
                                  FF5C  31    19E9  1933  BRW 80$
                          F014 CF   01    CA  19EC  1934 145$: BICL2 #1,NXM_RD_FLAG  ; Clear flag
                                 000019F1  19F1  1935  ENDTST = .
                                 FFFFFF80  19F1  1936  .IF GE<ENDTST-BGNTST- X300> ; Check test length
                                           19F1  1937  .ERROR ; MAXIMUM TEST LENGTH OF 300 HEX BYTES EXCEEDED!
                                           19F1  1938  .ENDC    ; End conditional
```

```
                19F1  1940  .SBTTL TEST 06:  TAG STORE DUAL ADDRESS TEST
                19F1  1941
                19F1  1942 ;++
                19F1  1943 ; FUNCTIONAL DESCRIPTION:
                19F1  1944 ;
                19F1  1945 ; This test checks the Index lines to the Cache Tag Store Rams
                19F1  1946 ; for bits S-A-0, S-A-1, and shorted together, by referencing
                19F1  1947 ; 12 Index locations generated by floating a one across the Index
                19F1  1948 ; (PAD<11:02>). A Tag Store value of zero is written at Index loc.
                19F1  1949 ; zero, and then a Tag Store value of EFF (as many bits set as
                19F1  1950 ; possible) is written at the next Index location (004). Index loc.
                19F1  1951 ; zero is then read again and checked for a hit.  This process
                19F1  1952 ; continues for writing zeroes each time at Index location zero and
                19F1  1953 ; then writing EFF into the next Index location. After having written
                19F1  1954 ; EFF at the other Index locations, zeroes are then written into Index
                19F1  1955 ; location 004, and EFF will be written into the Tag Store at all
                19F1  1956 ; the others. The sequence is controlled by an inner loop pointer
                19F1  1957 ; and counter for writing EFF into the Index locations, and an
                19F1  1958 ; outer loop pointer and counter to control writing zeroes into
                19F1  1959 ; the  Index locations.
                19F1  1960 ;
                19F1  1961 ; In order to check all Index lines, two subtests are needed:
                19F1  1962 ; the first subtest will use Tag Store values of 000 and EFF,
                19F1  1963 ; the second subtest will use Tag Store values of 000 and 100.
                19F1  1964 ;
                19F1  1965 ; NOTE:    In this test, a write reference to address EFF000
                19F1  1966 ; will always cause an expected Write NXM Interrupt to occur.
                19F1  1967 ; However, write references to address 100000 may or may not
                19F1  1968 ; be non-existent. This situation will be handled by setting the
                19F1  1969 ; NXM_WT_FLAG before writing to address 100000 and not testing
                19F1  1970 ; the flag after the reference. If the reference is a valid
                19F1  1971 ; PA no interrupt will occur, and the test will continue. If,
                19F1  1972 ; on the other hand, the reference to address 100000 is non-
                19F1  1973 ; existent, a Write NXM Interrupt will occur. The service
                19F1  1974 ; routine will first check that the NXM_WT_FLAG is set and
                19F1  1975 ; then will clear it and return to the test flow, where the
                19F1  1976 ; flag will not be tested.
                19F1  1977 ;
                19F1  1978 ;
                19F1  1979 ; IMPLICIT INPUTS:
                19F1  1980 ;
                19F1  1981 ; The following table of Index values will be used:
                19F1  1982 ;
                19F1  1983 ; CA_INDX_VAL:: .LONG   000
                19F1  1984 ;    .LONG    004
                19F1  1985 ;    .LONG    008
                19F1  1986 ;    .LONG   X010
                19F1  1987 ;    .LONG   X020
                19F1  1988 ;    .LONG   X040
                19F1  1989 ;    .LONG   X080
                19F1  1990 ;    .LONG   X100
                19F1  1991 ;    .LONG   X200
                19F1  1992 ;    .LONG   X400
                19F1  1993 ;    .LONG   X800
                19F1  1994 ;
                19F1  1995 ;
                19F1  1996 ;
```

```
                                    19F1  1997 ;--
                                    19F1  1998
                                    19F1  1999
                                    19F1  2000
                                    19F1  2001
                                    19F1  2002
                                    19F1  2003  SEQ  $TN    ; Macro call for Test Sequence Check
                                    19F1        TEST_6::    ; Continued on next line
          0000FE50'EF    06   B0   19F1         MOVW #6,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF   B6   19F8         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF   B1   19FE         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        1A04          APT_TEST_COUNT  ; Check for the correct test number
                     20   13   1A09         BEQL T6_S1    ; Sequence correct, continue test
          0000FE00 EF    1F   E1   1A0B         BBC #31,APT_CTRL_FLAGS,-
                     17        1A12          T6_HLT    ; Branch if not APT mode
     0000FE44 EF   00001A2A EF   9E   1A13         MOVAB T6_HLT,APT_ERR_PC ; Save the error PC
          0000FE40'EF    01   B0   1A1E         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                   F25A   30   1A25         BSBW APT_COMMAND  ; Call APT routine
                     01   11   1A28         BRB T6_S1    ; Continue
                                    1A2A        T6_HLT:
                          00   1A2A         HALT     ; ERROR!!! Program out of sequence
                                    1A2B        T6_S1:
           EFD6 CF    FD AF   DE   1A2B         MOVAL T6_S1,RET_PC  ; Store return PC
                                    1A31
                    0000   31   1A31  2004  BRW 5$    ; Go to start of test
                     00000A34  1A34  2005  TEMP = . & ^XFFF  ; Put Index value into TEMP
                     00000830  1A34  2006  .IF LE <TEMP- X204>  ; If Index is LE 204, make it 204.
                               1A34  2007  . = . + < X204-TEMP>  ; Make Index equal to 204
                               1A34  2008  .IFF     ; If false, go to next conditional
                     00000630  1A34  2009  .IF LE <TEMP- X404>  ; If Index is LE 404, make it 404
                               1A34  2010  . = . + < X404-TEMP>  ; Make Index equal to 404
                               1A34  2011  .IFF     ; If false, to to next conditional
                     0000042F  1A34  2012  .IF LT <TEMP- X605>  ; If Index LT 605, do not change.
                               1A34  2013  .IFF     ; If false, go to next conditional.
                     FFFFFC30  1A34  2014  .IF GT <TEMP- XE04>  ; If Index is GT E04, make is 204.
                               1A34  2015  . = <. & ^XFFFFF000> + ^X1204 ; Bump up Index to 204.
                               1A34  2016  .IFF     ; If false, to to next conditional.
                     00000230  1A34  2017  .IF LT <TEMP- X804>  ; If Index is LT 804, make it 804.
                               1A34  2018  . = . + < X804-TEMP>  ; Make Index equal to 804.
                               1A34  2019  .ENDC     ; End conditional
                               1A34  2020  .ENDC     ; End conditional
                               1A34  2021  .ENDC     ; End conditional
                               1A34  2022  .ENDC     ; End conditional
                               1A34  2023  .ENDC     ; End conditional
                     00001A34  1A34  2024  BGNTST = .
               25    00   DA   1A34  2025 5$: MTPR #0,#PR$_CADR  ; Flush the Cache
               27    0F   DA   1A37  2026  MTPR #^XF,#PR$_CAER  ; Clear the CAER
               54    01   D0   1A3A  2027  MOVL #1,R4    ; Initialize outer loop counter
          56   E85B CF    DE   1A3D  2028  MOVAL CA_INDX_VAL,R6  ; Set up Cache Index Value pointer
                               1A42  2029       ; for outer loop
               55    01   D0   1A42  2030 10$: MOVL #1,R5    ; Initialize inner loop counter
          57   E853 CF    DE   1A45  2031  MOVAL CA_INDX_VAL,R7  ; Set up Cache Index Value pointer
                               1A4A  2032       ; for inner loop
               67    66   D1   1A4A  2033 20$: CMPL (R6),(R7)  ; Make sure Index values are not equal
                     75   13   1A4D  2034  BEQL 70$    ; Get another Index val for inner loop
          FFB0 CF    01   8A   1A4F  2035  BICB2 #1,NXM_WT_FLAG  ; Clear NXM Write Flag
               58    66   D0   1A54  2036  MOVL (R6),R8    ; Store Index value
```

```
                00FFF460'EF    B5  1A57  2037  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                        68     D5  1A5D  2038  TSTL (R8)   ; Write zeroes into Tag Store spec.
                                   1A5F  2039       ; by R8.
                     50   67    D0  1A5F  2040  MOVL (R7),R0   ; Store next Index value into R0
           50    00EFF000 8F    C0  1A62  2041  ADDL2 # XEFF000,R0  ; Generate next Tag Store address
               EF96 CF    01    88  1A69  2042 30$: BISB2 #1,NXM_WT_FLAG  ; Set flag for expected write to NXM
                00FFF460'EF    B5  1A6E  2043  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                        60     D4  1A74  2044  CLRL (R0)   ; Write to Tag Store address spec.
                                   1A76  2045       ; by R0
                                01 1A76  2046  NOP    ; Wait for interrupt to occur
             1F EF88 CF    00    E1  1A77  2047  BBC #0,NXM_WT_FLAG,50$ ; If interrupt occured, continue test
          14 0000FE00'EF    1F    E1  1A7D  2048  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
          0000FE44 EF    99'AF    3E  1A85  2049  MOVAW B 40$,APT_ERR_PC ; Save Error PC
             0000FE40'EF    01    88  1A8D  2050  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         F1EB    30  1A94  2051  BSBW APT_COMMAND
                           03    11  1A97  2052  BRB 50$
                           00    00  1A99  2053 40$: HALT    ; ERROR, NXM Write interrupt did not
                                   1A9A  2054       ; occur.
                                   1A9A  2055       ; R0 contains referenced address
                           CD    11  1A9A  2056  BRB 30$   ; Loop on error
                           68    D5  1A9C  2057 50$: TSTL (R8)   ; Read previous Index location
                                   1A9E  2058       ; should cause Read Hit
                        52   27    DB  1A9E  2059  MFPR #CAER,R2  ; Store hit/miss information
                     1F 52   00    E0  1AA1  2060  BBS #0,R2,70$  ; If Cache hit, continue test
          14 0000FE00'EF    1F    E1  1AA5  2061  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
          0000FE44 EF    C1'AF    3E  1AAD  2062  MOVAW B 60$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88  1AB5  2063  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         F1C3    30  1ABC  2064  BSBW APT_COMMAND
                           03    11  1ABF  2065  BRB 70$
                           00    00  1AC1  2066 60$: HALT    ; ERROR, did not get Cache hit
                                   1AC2  2067       ; R8 contains referenced address
                                   1AC2  2068       ; R0 contains previously ref. address
                                   1AC2  2069       ; R2 contains contents of CAER
                           A5    11  1AC2  2070  BRB 30$   ; Loop on error
                     25    00    DA  1AC4  2071 70$: MTPR #0,#CADR  ; Flush the Cache
                     57    04    C0  1AC7  2072  ADDL2 #4,R7   ; Point to next Index Val inner loop
                           55    D6  1ACA  2073  INCL R5
                     0B    55    91  1ACC  2074  CMPB R5,#11
                           03    14  1ACF  2075  BGTR 75$
                        FF76    31  1AD1  2076  BRW 20$
                     56    04    C0  1AD4  2077 75$: ADDL2 #4,R6   ; Point to next Index Val outer loop.
          FF65 54    01    0B    9D  1AD7  2078  ACBB #11,#1,R4,10$  ; Continue outer loop for 11 Indx val.
                                   1ADD  2079
            EF24 CF    FD AF    DE  1ADD  2080 T6_S2: MOVAL T6_S2,RET_PC  ; Save return PC
                     25    00    DA  1AE3  2081  MTPR #0,#CADR  ; Flush the Cache
                     27    0F    DA  1AE6  2082  MTPR # XF,#CAER  ; Clear the CAER
                     54    01    D0  1AE9  2083  MOVL #1,R4   ; Initialize outer loop counter
                56   E7AC CF    DE  1AEC  2084  MOVAL CA_INDX_VAL,R6  ; Set up Index Val pointer, outer loop
                     55    01    D0  1AF1  2085 10$: MOVL #1,R5   ; Initialize inner loop counter
                57   E7A4 CF    DE  1AF4  2086  MOVAL CA_INDX_VAL,R7  ; Set up Index Val pointer, inner loop
                     67    66    D1  1AF9  2087 20$: CMPL (R6),(R7)  ; Make sure Index values are not equal
                           50    13  1AFC  2088  BEQL 60$   ; Get another Index val for inner loop
                EF01 CF    01    8A  1AFE  2089  BICB2 #1,NXM_WT_FLAG  ; Clear NXM Write Flag
                     58    66    D0  1B03  2090  MOVL (R6),R8   ; Store Cache Index val in R8
                 00FFF460 EF    B5  1B06  2091  TSTW SYNC_POINT  ; Provide Sync on Unibus Master Sync
                           68    D5  1B0C  2092  TSTL (R8)   ; Write 0 s into CA Tag Store spec.
                                   1B0E  2093       ; by R8.
```

```
                50    67    D0  1B0E  2094  MOVL (R7),R0    ; Store next Index value into R0
           50   00100000 8F    C0  1B11  2095  ADDL2 #X100000,R0   ; Generate next Tag Store Address
               EEE7 CF    01    88  1B18  2096 30$: BISB2 #1,NXM_WT_FLAG   ; Set NXM WT FLAG
                00FFF460'EF    B5  1B1D  2097  TSTW SYNC_POINT   ; Provide Sync on Unibus Master Sync
                       60    D4  1B23  2098  CLRL (R0)    ; Write to Tag Store Addr. spec. by R0
                       01    1B25  2099  NOP     ; Wait in event of Interrupt
                       68    D5  1B26  2100 40$: TSTL (R8)    ; Read previous Index location
                   52  27    DB  1B28  2101  MFPR #CAER,R2   ; Store hit/miss information
                 1F 52  00    E0  1B2B  2102  BBS #0,R2,60$   ; If Cache hit, go to next test
          14 0000FE00 EF  1F    E1  1B2F  2103  BBC #31,APT_CTRL_FLAGS,50$ ; Branch if not APT mode
          0000FE44'EF   4B'AF    3E  1B37  2104  MOVAW B 50$,APT_ERR_PC ; Save Error PC
             0000FE40'EF  01    88  1B3F  2105  BISB2 #1,APT_ERR_FLAG   ; Set Error Flag
                     F139  30    1B46  2106  BSBW APT_COMMAND
                       03    11  1B49  2107  BRB 60$
                             00  1B4B  2108 50$: HALT     ; ERROR, Did not get Cache hit
                                 1B4C  2109       ; R8 contains referenced address
                                 1B4C  2110       ; R0 contains previously ref. address.
                                 1B4C  2111       ; R2 contains contents of CAER
                       CA    11  1B4C  2112  BRB 30$    ; Loop on error
                   25  00    DA  1B4E  2113 60$: MTPR #0,#CADR   ; Flush the Cache
                   57  04    C0  1B51  2114  ADDL2 #4,R7    ; Point to next Index val. inner loop
                A1 55  0B    F3  1B54  2115  AOBLEQ #11,R5,20$   ; Continue inner loop for 11 Index val
                   56  04    C0  1B58  2116  ADDL2 #4,R6    ; Point to next Index val. outer loop
                92 54  0B    F3  1B5B  2117  AOBLEQ #11,R4,10$   ; Continue outer loop for 11 Index val
                         00001B5F  1B5F  2118  ENDTST = .
                         FFFFFF3B  1B5F  2119  .IF GE<ENDTST-BGNTST- X1F0> ; Check test length
                                   1B5F  2120  .ERROR ; MAXIMUM TEST LENGTH OF 1F0 HEX BYTES EXCEEDED!
                                   1B5F  2121  .ENDC     ; End conditional
```

```
                     1B5F  2123  .SBTTL TEST 07:  DATA STORE DUAL ADDRESS TEST
                     1B5F  2124
                     1B5F  2125 ;++
                     1B5F  2126 ; FUNCTIONAL DESCRIPTION:
                     1B5F  2127 ;
                     1B5F  2128 ; This test checks the Index lines to the Cache Data Store
                     1B5F  2129 ; Rams for bits S-A-0, S-A-1, and shorted together by
                     1B5F  2130 ; floating a one across a field of zeroess in the Index lines
                     1B5F  2131 ; of the Data Store and checking for correct data. A Data
                     1B5F  2132 ; Store value of zero is written at Index locaion zero, then
                     1B5F  2133 ; all ones are written into the Data Store at Index location
                     1B5F  2134 ; 004. Index location zero is then read and checked for the
                     1B5F  2135 ; correct data off all zeroes. The process is repeated for 12
                     1B5F  2136 ; Index patterns in the same manner as used in the Cache Tag
                     1B5F  2137 ; Store Test.
                     1B5F  2138 ;
                     1B5F  2139 ;
                     1B5F  2140 ;
                     1B5F  2141 ;
                     1B5F  2142 ;
                     1B5F  2143 ; IMPLICIT INPUTS:
                     1B5F  2144 ;
                     1B5F  2145 ; The following table of Index values will be used:
                     1B5F  2146 ;
                     1B5F  2147 ; CA_INDX_VAL:: .LONG   000
                     1B5F  2148 ;    .LONG   004
                     1B5F  2149 ;    .LONG   008
                     1B5F  2150 ;    .LONG  X010
                     1B5F  2151 ;    .LONG  X020
                     1B5F  2152 ;    .LONG  X040
                     1B5F  2153 ;    .LONG  X080
                     1B5F  2154 ;    .LONG  X100
                     1B5F  2155 ;    .LONG  X200
                     1B5F  2156 ;    .LONG  X400
                     1B5F  2157 ;    .LONG  X800
                     1B5F  2158 ;
                     1B5F  2159 ;
                     1B5F  2160 ;
                     1B5F  2161 ;--
                     1B5F  2162
                     1B5F  2163
                     1B5F  2164
                     1B5F  2165
                     1B5F  2166
                     1B5F  2167
                     1B5F  2168  SEQ  $TN    ; Macro call for Test Sequence Check
                     1B5F        TEST_7::    ; Continued on next line
0000FE50'EF   07   B0  1B5F         MOVW #7,APT_TEST_NUM ; Load the Test Number
     0000FE52'EF   B6  1B66         INCW APT_TEST_COUNT  ; Bump up the test count
     0000FE50'EF   B1  1B6C         CMPW APT_TEST_NUM,-  ; Continued on next line
     0000FE52'EF       1B72          APT_TEST_COUNT  ; Check for the correct test number
                20 13  1B77         BEQL T7_S1    ; Sequence correct, continue test
0000FE00 EF     1F E1  1B79         BBC #31,APT_CTRL_FLAGS,-
                17     1B80          T7_HLT    ; Branch if not APT mode
0000FE44'EF   00001B98 EF  9E  1B81         MOVAB T7_HLT,APT_ERR_PC ; Save the error PC
0000FE40'EF   01   B0  1B8C         MOVW #1,APT_ERR_FLAG  ; Set the error flag
            F0EC   30  1B93         BSBW APT_COMMAND  ; Call APT routine
```

```
                    01   11  1B96          BRB T7_S1    ; Continue
                                1B98        T7_HLT:
                         00     1B98          HALT     ; ERROR!!! Program out of sequence
                                1B99        T7_S1:
          EE68 CF   FD AF   DE  1B99          MOVAL T7_S1,RET_PC  ; Store return PC
                                1B9F
                  0000   31     1B9F  2169    BRW 5$    ; Go to start of test
                   00000BA2     1BA2  2170    TEMP = . & ^XFFF  ; Put Index value into TEMP
                   0000099E     1BA2  2171    .IF LE <TEMP- X204>  ; If Index is LE 204, make it 204.
                                1BA2  2172    . = . + < X204-TEMP>  ; Make Index equal to 204
                                1BA2  2173    .IFF     ; If false, go to next conditional
                   0000079E     1BA2  2174    .IF LE <TEMP- X404>  ; If Index is LE 404, make it 404
                                1BA2  2175    . = . + < X404-TEMP>  ; Make Index equal to 404
                                1BA2  2176    .IFF     ; If false, to to next conditional
                   0000059D     1BA2  2177    .IF LT <TEMP- X605>  ; If Index LT 605, do not change.
                                1BA2  2178    .IFF     ; If false, go to next conditional.
                   FFFFFD9E     1BA2  2179    .IF GT <TEMP- XE04>  ; If Index is GT E04, make is 204.
                                1BA2  2180    . = <. & ^XFFFFF000> + ^X1204 ; Bump up Index to 204.
                                1BA2  2181    .IFF     ; If false, to to next conditional.
                   0000039E     1BA2  2182    .IF LT <TEMP- X804>  ; If Index is LT 804, make it 804.
                                1BA2  2183    . = . + < X804-TEMP>  ; Make Index equal to 804.
                                1BA2  2184    .ENDC     ; End conditional
                                1BA2  2185    .ENDC     ; End conditional
                                1BA2  2186    .ENDC     ; End conditional
                                1BA2  2187    .ENDC     ; End conditional
                                1BA2  2188    .ENDC     ; End conditional
                   00001BA2     1BA2  2189    BGNTST = .
              25    00   DA     1BA2  2190 5$: MTPR #0,#CADR  ; Flush the Cache
              27    0F   DA     1BA5  2191    MTPR # ^XF,#CAER  ; Clear the CAER
              54    01   D0     1BA8  2192    MOVL #1,R4    ; Initialize outer loop counter
         56   E6ED CF   DE     1BAB  2193    MOVAL CA_INDX_VAL,R6  ; Set up Cache Index Value pointer
                                1BB0  2194          ; for outer loop.
              55    01   D0     1BB0  2195 10$: MOVL #1,R5    ; Initialize inner loop counter
         57   E6E5 CF   DE     1BB3  2196    MOVAL CA_INDX_VAL,R7  ; Set up Cache Index Value pointer
                                1BB8  2197          ; for inner loop.
              67    66   D1     1BB8  2198 20$: CMPL (R6),(R7)  ; Make sure Index values are not equal
                    48   13     1BBB  2199    BEQL 50$    ; Get another Index val for inner loop
         EE42 CF    01   8A     1BBD  2200    BICB2 #1,NXM_WT_FLAG  ; Clear flag
              58    66   D0     1BC2  2201    MOVL (R6),R8    ; Store Index value
    58    00010000   8F   C0     1BC5  2202    ADDL2 # X10000,R8  ; Make references away from test code.
                    68   D4     1BCC  2203 25$: CLRL (R8)    ; Write zeroes into Data Store
                                1BCE  2204          ; at Index loc. spec by R8
              50    67   D0     1BCE  2205    MOVL (R7),R0    ; Store next Index value
    50    00010000   8F   C0     1BD1  2206    ADDL2 # X10000,R0  ; Make references away from test code.
    60    FFFFFFFF   8F   D0     1BD8  2207    MOVL # ^XFFFFFFFF,(R0) ; Write ones into Data Store
                                1BDF  2208          ; at Index loc. spec. by R0
              53    68   D0     1BDF  2209 30$: MOVL (R8),R3    ; Read Cache Data Store at Index loc.
                                1BE2  2210          ; Specified by R8
                    53   D5     1BE2  2211    TSTL R3    ; Check data read from Cache
                    1F   13     1BE4  2212    BEQL 50$    ; If data correct, continue test
14 0000FE00 EF      1F   E1     1BE6  2213    BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
0000FE44 EF    02   AF   3E     1BEE  2214    MOVAW B 40$,APT_ERR_PC ; Save Error PC
   0000FE40 EF      01   88     1BF6  2215    BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                  F082   30     1BFD  2216    BSBW APT_COMMAND
                    03   11     1C00  2217    BRB 50$
                         00     1C02  2218 40$: HALT     ; ERROR, Data read in Cache incorrect
                                1C03  2219          ; R8 contains referenced address
```

```
                                1C03  2220       ; Data read should be all zeroes
                                1C03  2221       ; R3 contains expected data
                      C7   11   1C03  2222  BRB 25$    ; Loop on error
                  25  00   DA   1C05  2223 50$: MTPR #0,#CADR  ; Flush the Cache
                  57  04   C0   1C08  2224  ADDL2 #4,R7    ; Point to next Index Val inner loop
               A9 55  0B   F3   1C0B  2225  AOBLEQ #11,R5,20$  ; Continue inner loop for 11 Indx val
                  56  04   C0   1C0F  2226  ADDL2 #4,R6    ; Point to next Index Val outer loop
               9A 54  0B   F3   1C12  2227  AOBLEQ #11,R4,10$  ; Continue outer loop for 11 Indx val
                                1C16  2228
                     0000   31   1C16  2229  BRW TEST_8    ; Go to next test
                      00001C19  1C19  2230  ENDTST = .
                      FFFFFF77  1C19  2231  .IF GE<ENDTST-BGNTST- X100> ; Check test length
                                1C19  2232  .ERROR ; MAXIMUM TEST LENGTH OF 100 HEX BYTES EXCEEDED!
                                1C19  2233  .ENDC     ; End conditional
```

```
                        1C19  2235  .SBTTL TEST 08:  BYTE/WORD WRITE MISS TEST
                        1C19  2236
                        1C19  2237 ;++
                        1C19  2238 ; FUNCTIONAL DESCRIPTION:
                        1C19  2239 ;
                        1C19  2240 ; This test checks that a byte or word write miss will not
                        1C19  2241 ; get cached.
                        1C19  2242 ;
                        1C19  2243 ; The code first writes a longword of zeroes in Cache
                        1C19  2244 ; at Test location 10000. A byte of ones is then written
                        1C19  2245 ; at Test location 11000, which is the same Index position
                        1C19  2246 ; as 10000. However, since this is a byte write, the reference
                        1C19  2247 ; should not be written in Cache. Read reference is then made
                        1C19  2248 ; to location 10000 and its checked for a hit and correct data.
                        1C19  2249 ; This indicates that the original write of zeroes did not
                        1C19  2250 ; get overwritten.
                        1C19  2251 ; The test is repeated performing a word write instead of byte
                        1C19  2252 ; write.
                        1C19  2253 ;
                        1C19  2254 ;
                        1C19  2255 ;--
                        1C19  2256
                        1C19  2257
                        1C19  2258  SEQ  $TN    ; Macro call for Test Sequence Check
                        1C19        TEST_8::    ; Continued on next line
0000FE50'EF   08   B0   1C19         MOVW #8,APT_TEST_NUM ; Load the Test Number
     0000FE52'EF   B6   1C20         INCW APT_TEST_COUNT  ; Bump up the test count
     0000FE50'EF   B1   1C26         CMPW APT_TEST_NUM,-  ; Continued on next line
     0000FE52 EF        1C2C          APT_TEST_COUNT  ; Check for the correct test number
                20   13   1C31         BEQL T8_S1    ; Sequence correct, continue test
0000FE00'EF   1F   E1   1C33         BBC #31,APT_CTRL_FLAGS,-
                17      1C3A          T8_HLT    ; Branch if not APT mode
0000FE44 EF   00001C52'EF   9E   1C3B         MOVAB T8_HLT,APT_ERR_PC ; Save the error PC
0000FE40'EF   01   B0   1C46         MOVW #1,APT_ERR_FLAG  ; Set the error flag
           F032   30   1C4D         BSBW APT_COMMAND  ; Call APT routine
             01   11   1C50         BRB T8_S1    ; Continue
                       1C52        T8_HLT:
                00     1C52         HALT     ; ERROR!!! Program out of sequence
                       1C53        T8_S1:
 EDAE CF   FD AF   DE   1C53         MOVAL T8_S1,RET_PC  ; Store return PC
                       1C59
              0000   31   1C59  2259  BRW 5$    ; Go to start of test
               00000C5C  1C5C  2260  TEMP = . & ^XFFF  ; Put present Index value into TEMP
               00000C58  1C5C  2261  .IF LE<TEMP-004>  ; If Index LE 004, make it 004
                         1C5C  2262  . = . + <004-TEMP>  ; Make Index value equal to 004
                         1C5C  2263  .IFF     ; If false, go to next conditional
               FFFFFEC0  1C5C  2264  .IF GT<TEMP- ^XE20>  ; If Index GT E20, move up to 004
                         1C5C  2265  . = <. & ^XFFFF000> + ^X1004 ; Move Index up to next 004
                         1C5C  2266  .ENDC     ; End conditional
                         1C5C  2267  .ENDC     ; End conditional
               00001C5C  1C5C  2268  BGNTST = .
          25   00   DA   1C5C  2269 5$: MTPR #0,#PR$_CADR  ; Flush the Cache
          27   0F   DA   1C5F  2270  MTPR #^XF,#PR$_CAER  ; Clear the CAER
50   00010000 8F   D0   1C62  2271  MOVL #^X10000,R0  ; Load Test location 1
51   00011000 8F   D0   1C69  2272  MOVL #^X11000,R1  ; Load Test location 2
                60   D4   1C70  2273 10$: CLRL (R0)    ; Write zeroes into Test loc.1 (10000)
           61   FF 8F   90   1C72  2274  MOVB #^XFF,(R1)  ; Write a byte of ones into Test loc 2
```

```
                    52   60    D0  1C76  2275  MOVL (R0),R2   ; Read Test location 1 (10000)
                    53   27    DB  1C79  2276  MFPR #CAER,R3  ; Store hit/miss information
                 20 53   00    E0  1C7C  2277  BBS #0,R3,30$  ; If Cache hit, continue test
           14 0000FE00 EF   1F    E1  1C80  2278  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
           0000FE44 EF   9C'AF    3E  1C88  2279  MOVAW B 20$,APT_ERR_PC ; Save Error PC
              0000FE40'EF   01    88  1C90  2280  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          EFE8    30  1C97  2281  BSBW APT_COMMAND
                            04    11  1C9A  2282  BRB 30$
                                  00  1C9C  2283 20$: HALT    ; ERROR, did not get Cache hit when
                                      1C9D  2284       ; reading Test location 1 (10000)
                                      1C9D  2285       ; R0 contains address read
                                      1C9D  2286       ; R1 contains byte write address
                                      1C9D  2287       ; R3 contains contents of CAER
                          FFD0    31  1C9D  2288  BRW 10$   ; Loop on error
                            52    D5  1CA0  2289 30$: TSTL R2   ; Check for correct data results
                            20    13  1CA2  2290  BEQL T8_S2   ; If data correct go to next subtest
           14 0000FE00 EF   1F    E1  1CA4  2291  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
           0000FE44'EF   C0'AF    3E  1CAC  2292  MOVAW B 40$,APT_ERR_PC ; Save Error PC
              0000FE40'EF   01    88  1CB4  2293  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          EFC4    30  1CBB  2294  BSBW APT_COMMAND
                            04    11  1CBE  2295  BRB 50$
                                  00  1CC0  2296 40$: HALT    ; ERROR, data incorrect
                                      1CC1  2297       ; expected data = 00000000
                                      1CC1  2298       ; R2 contains received data
                          FFAC    31  1CC1  2299  BRW 10$   ; Loop on error
                                      1CC4  2300 50$:
                ED3D CF   FD AF    DE  1CC4  2301 T8_S2: MOVAL T8_S2,RET_PC  ; Store return PC
                       25   00    DA  1CCA  2302  MTPR #0,#CADR  ; Flush the Cache
                       27   0F    DA  1CCD  2303  MTPR # XF,#CAER  ; Clear the CAER
                50   00010000 8F    D0  1CD0  2304  MOVL # X10000,R0  ; Load Test location 1
                51   00011000 8F    D0  1CD7  2305  MOVL # X11000,R1  ; Load Test location 2
                             60    D4  1CDE  2306 10$: CLRL (R0)   ; Write zeroes into Test loc.1 (10000)
                    61   FFFF 8F    B0  1CE0  2307  MOVW # XFFFF,(R1)  ; Write a word of ones into Test loc 2
                       52   60    D0  1CE5  2308  MOVL (R0),R2   ; Read Test location 1 (10000)
                       53   27    DB  1CE8  2309  MFPR #CAER,R3  ; Store hit/miss information
                    1F 53   00    E0  1CEB  2310  BBS #0,R3,30$  ; If Cache hit, continue test
           14 0000FE00 EF   1F    E1  1CEF  2311  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
           0000FE44'EF   0B'AF    3E  1CF7  2312  MOVAW B 20$,APT_ERR_PC ; Save Error PC
              0000FE40 EF   01    88  1CFF  2313  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          EF79    30  1D06  2314  BSBW APT_COMMAND
                            03    11  1D09  2315  BRB 30$
                                  00  1D0B  2316 20$: HALT    ; ERROR, did not get Cache hit when
                                      1DCC  2317       ; reading Test location 1 (10000)
                                      1D0C  2318       ; R0 contains address read
                                      1D0C  2319       ; R1 contains word write address
                                      1D0C  2320       ; R3 contains contents of CAER
                            D0    11  1D0C  2321  BRB 10$   ; Loop on error
                            52    D5  1D0E  2322 30$: TSTL R2   ; Check for correct data results
                            1F    13  1D10  2323  BEQL TEST_9   ; If data correct go to next subtest
           14 0000FE00 EF   1F    E1  1D12  2324  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
           0000FE44 EF   2E AF    3E  1D1A  2325  MOVAW B 40$,APT_ERR_PC ; Save Error PC
              0000FE40 EF   01    88  1D22  2326  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          EF56    30  1D29  2327  BSBW APT_COMMAND
                            03    11  1D2C  2328  BRB 50$
                                  00  1D2E  2329 40$: HALT    ; ERROR, data incorrect
                                      1D2F  2330       ; expected data = 00000000
                                      1D2F  2331       ; R2 contains received data
```

```
                AD   11  1D2F  2332  BRB 10$   ; Loop on error
                         1D31  2333 50$:
            00001D31     1D31  2334  ENDTST = .
            FFFFFF85     1D31  2335  .IF GE<ENDTST-BGNTST- X150> ; Check test length
                         1D31  2336  .ERROR ; MAXIMUM TEST LENGTH OF 150 HEX BYTES EXCEEDED!
                         1D31  2337  .ENDC    ; End conditional
```

```
                     1D31  2339  .SBTTL  TEST 09:  CACHE TAG STORE DYNAMIC MEMORY TEST
                     1D31  2340 ;++
                     1D31  2341 ; FUNCTIONAL DESCRIPTION:
                     1D31  2342 ;
                     1D31  2343 ; This test verifies the ability to write and read all of the
                     1D31  2344 ; locations in the Cache Tag Store Rams by use of a marching
                     1D31  2345 ; pattern applied across PAD<23:12>. Two Tag Store addresses
                     1D31  2346 ; are used in this test:
                     1D31  2347 ;
                     1D31  2348 ; Cache Tag Store address 1 = 100 hex
                     1D31  2349 ; Cache Tag Store address 2 = EFF hex
                     1D31  2350 ;
                     1D31  2351 ; The routine initially writes all the Cache Tag Store with
                     1D31  2352 ; address 1 (100).  Address 1 is then checked at the first
                     1D31  2353 ; Index location, and its compliment (EFF) is written. This
                     1D31  2354 ; type of sequence increments and decrements throughout all
                     1D31  2355 ; Cache Index locations.
                     1D31  2356 ;
                     1D31  2357 ; The test is divided into four subtests.
                     1D31  2358 ;
                     1D31  2359 ; Subtest 1 checks the first half of the Tag Store while
                     1D31  2360 ; incrementing through the Index (000-7FC).
                     1D31  2361 ;
                     1D31  2362 ; Subtest 2 checks the first half of the Tag Store while
                     1D31  2363 ; decrementing through the Index (7FC-000).
                     1D31  2364 ;
                     1D31  2365 ; Subtest 3 checks the second half of the Tag Store while
                     1D31  2366 ; incrementing through the Index (800-FFC).
                     1D31  2367 ;
                     1D31  2368 ; Subtest 4 checks the second half of the Tag Store while
                     1D31  2369 ; decrementing through the Index (FFC-800).
                     1D31  2370 ;
                     1D31  2371 ; NOTE: At the beginning of the test the IPL is raised to 1F
                     1D31  2372 ;  in order to block out Write NXM Interrupts that will
                     1D31  2373 ;  occur when referencing Tags of EFF000 and possibly
                     1D31  2374 ;  100000. This eliminates the necessity of making sure
                     1D31  2375 ;  that the SCB, SP, and WT NXM Service routines do not
                     1D31  2376 ;  cause Cache hits that reference Index locations under
                     1D31  2377 ;  test.
                     1D31  2378 ;
                     1D31  2379 ;
                     1D31  2380 ;
                     1D31  2381 ;
                     1D31  2382 ;--
                     1D31  2383
                     1D31  2384
                     1D31  2385  SEQ  $TN    ; Macro call for Test Sequence Check
                     1D31       TEST_9::     ; Continued on next line
          0000FE50 EF   09   B0  1D31         MOVW #9,APT_TEST_NUM ; Load the Test Number
               0000FE52 EF   B6  1D38         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50 EF   B1  1D3E         CMPW APT_TEST_NUM,   ; Continued on next line
               0000FE52 EF       1D44          APT_TEST_COUNT  ; Check for the correct test number
                        20   13  1D49         BEQL T9_S1    ; Sequence correct, continue test
          0000FE00 EF   1F   E1  1D4B         BBC #31,APT_CTRL_FLAGS,-
                        17       1D52          T9_HLT    ; Branch if not APT mode
0000FE44  EF   00001D6A EF   9E  1D53         MOVAB T9_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF   01   B0  1D5E         MOVW #1,APT_ERR_FLAG  ; Set the error flag
```

```
                    EF1A   30  1D65          BSBW APT_COMMAND  ; Call APT routine
                      01   11  1D68          BRB T9_S1   ; Continue
                           1D6A          T9_HLT:
                           00  1D6A          HALT    ; ERROR!!! Program out of sequence
                           1D6B          T9_S1:
          EC96 CF    FD AF   DE  1D6B          MOVAL T9_S1,RET_PC  ; Store return PC
                           1D71
                           1D71  2386
                           1D71  2387 ;
                           1D71  2388 ; The following code fills one half of the 1024 Cache Tag Store locations,
                           1D71  2389 ; (000-7FC), with 100 Hex.
                           1D71  2390 ;
                      0000   31  1D71  2391  BRW 5$   ; Go to start of test
                       00000D74  1D74  2392  TEMP = . & ^XFFF  ; Put Index value into TEMP
                       00000574  1D74  2393  .IF LT <TEMP- X800>  ; If Index LT 800, make 800
                           1D74  2394  . = . +< X800-TEMP>  ; Make Index value equal to 800
                           1D74  2395  .IFF    ; If false, go to next conditional
                       FFFFFF74  1D74  2396  .IF GT <TEMP- XE00>  ; If Index GT E00, make 800
                           1D74  2397  . = <. & ^XFFFFF000 + X1800> ; Bump up Index to 800
                           1D74  2398  .ENDC    ; End conditional
                           1D74  2399  .ENDC    ; End conditional
                       00001D74  1D74  2400  BGNTST = .
                 12    1F   DA  1D74  2401 5$: MTPR #^X1F,#IPL  ; Raise the IPL to block WT NXM Intrs.
                 25    00   DA  1D77  2402  MTPR #0,#CADR  ; Flush the Cache
                 27    0F   DA  1D7A  2403  MTPR #^XF,#CAER  ; Clear the CAER
                       54   D4  1D7D  2404  CLRL R4   ; Initialize Index Counter
          50   00100000 8F   D0  1D7F  2405  MOVL #^X100000,R0  ; Generate Cache Tag Store Addr. 1
                       80   D4  1D86  2406 10$: CLRL (R0)+   ; Write 100 into Cache Tag Store
       F6 54   000001FF 8F   F3  1D88  2407  AOBLEQ #^X1FF,R4,10$  ; Continue for 512 Index locations
                           1D90  2408
                           1D90  2409 ;
                           1D90  2410 ; The next section of code reads and checks each Tag Store location
                           1D90  2411 ; for a hit (tag=100), and then writes the complement into that same
                           1D90  2412 ; location (tag=EFF). This continues for incrementing through the first
                           1D90  2413 ; 512 locations.
                           1D90  2414 ;
                           1D90  2415
                       54   D4  1D90  2416  CLRL R4   ; Initialize Index Counter
          50   00100000 8F   D0  1D92  2417  MOVL #^X100000,R0  ; Generate Cache Tag Store Addr. 1
          51   00EFF000 8F   D0  1D99  2418  MOVL #^XEFF000,R1  ; Generate Cache Tag Store Addr. 2
                       60   94  1DA0  2419 20$: CLRB (R0)   ; Read Tag Store Address 1 (100)
                    53   27  DB  1DA2  2420  MFPR #CAER,R3  ; Store hit/miss information
                 1F 53   00  E0  1DA5  2421  BBS #0,R3,40$  ; If Cache hit, continue test
       14 0000FE00 EF   1F  E1  1DA9  2422  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
       0000FE44 EF    C5 AF  3E  1DB1  2423  MOVAW B 30$,APT_ERR_PC ; Save Error PC
          0000FE40 EF   01  88  1DB9  2424  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      EEBF  30  1DC0  2425  BSBW APT_COMMAND
                       03   11  1DC3  2426  BRB 40$
                           00  1DC5  2427 30$: HALT    ; ERROR, did not get Cache hit
                           1DC6  2428      ; R0 contains referenced address
                       D8   11  1DC6  2429  BRB 20$   ; Loop on error
                       61   D4  1DC8  2430 40$: CLRL (R1)   ; Write EFF into Cache Tag Store
                    50   04  C0  1DCA  2431  ADDL2 #4,R0   ; Generate next Index value for addr 1
                    51   04  C0  1DCD  2432  ADDL2 #4,R1   ; Generate next Index value for addr 2
       C8 54   000001FF 8F   F3  1DD0  2433  AOBLEQ #^X1FF,R4,20$  ; Continue for 512 Index locations
                           1DD8  2434
                           1DD8  2435 ;
```

```
                                1DD8  2436 ; The next section of code reads and checks each Cache Tag Store
                                1DD8  2437 ; location for a hit (tag=EFF), and then writes the complement into
                                1DD8  2438 ; that same location (tag=100). This continues for incrementing through
                                1DD8  2439 ; the first 512 Index locations.
                                1DD8  2440 ;
                                1DD8  2441
                                1DD8  2442
                      54   D4   1DD8  2443  CLRL R4    ; Initialize Index Counter
     50   00100000 8F   D0   1DDA  2444  MOVL # X100000,R0  ; Generate Cache Tag Store Addr. 1
     51   00EFF000 8F   D0   1DE1  2445  MOVL # XEFF000,R1  ; Generate Cache Tag Store Addr. 2
                      61   94   1DE8  2446 50$: CLRB (R1)    ; Read Tag Store Address 2 (EFF)
                   53   27   DB   1DEA  2447  MFPR #CAER,R3   ; Store hit/miss information
                1F 53   00   E0   1DED  2448  BBS #0,R3,70$   ; If Cache hit, continue test
          14 0000FE00 EF   1F   E1   1DF1  2449  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
          0000FE44 EF    0D'AF   3E   1DF9  2450  MOVAW B 60$,APT_ERR_PC ; Save Error Pc
             0000FE40 EF   01   88   1E01  2451  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         EE77   30   1E08  2452  BSBW APT_COMMAND
                      03   11   1E0B  2453  BRB 70$
                      00   1E0D  2454 60$: HALT     ; ERROR, did not get Cache hit
                           1E0E  2455       ; R1 contains referenced address
                      D8   11   1E0E  2456  BRB 50$    ; Loop on error
                      60   D4   1E10  2457 70$: CLRL (R0)    ; Write 100 into Cache Tag Store
                   50   04   C0   1E12  2458  ADDL2 #4,R0    ; Generate next Index value for addr 1
                   51   04   C0   1E15  2459  ADDL2 #4,R1    ; Generate next Index value for addr 2
          C8 54    000001FF 8F   F3   1E18  2460  AOBLEQ # X1FF,R4,50$  ; Continue for 512 Index locations
                      1E   11   1E20  2461  BRB T9_S2    ; Go to next subtest
                           1E22  2462
                           1E22  2463 ;
                           1E22  2464 ; The next section of code is used only to initialize conditions to
                           1E22  2465 ; execute T9_S2 in event a scope loop is desired as a result of a
                           1E22  2466 ; Machine Check Exception. This code shortens the scope loop.
                           1E22  2467 ;
                           1E22  2468
                           1E22  2469 T9_INIT1:     ; Return label for T8_S2 loop on error
                   25   00   DA   1E22  2470  MTPR #0,#CADR   ; Flush the Cache
                   27   0F   DA   1E25  2471  MTPR # XF,#CAER   ; Clear the CAER
                      54   D4   1E28  2472  CLRL R4    ; Initialize Index Counter
          50   00100000 8F   D0   1E2A  2473  MOVL # X100000,R0  ; Generate Cache Tag Store Addr. 1
              EBCE CF   01   88   1E31  2474 10$: BISB2 #1,NXM_WT_FLAG  ; Set flag for expected NXM writes
                      80   D4   1E36  2475  CLRL (R0)+    ; Write 100 into Cache Tag Store
          F1 54    000001FF 8F   F3   1E38  2476  AOBLEQ # X1FF,R4,10$  ; Continue for 512 Index locations
                           1E40  2477
                           1E40  2478 ;
                           1E40  2479 ; The next section of code sequences from the largest Index value
                           1E40  2480 ; of the first half of the Tag Store to the smallest(7FC-000), each time
                           1E40  2481 ; reading and checking each Cache Tag Store location for a hit (tag=100),
                           1E40  2482 ; and then writing the complement into that same location (tag=EFF).
                           1E40  2483 ;
                           1E40  2484
           EBC1 CF    DF AF   DE   1E40  2485 T9_S2: MOVAL T9_INIT1,RET_PC  ; Store return PC to loop on T9_S2.
                      54   D4   1E46  2486  CLRL R4    ; Initialize Index Counter
          50   001007FC 8F   D0   1E48  2487  MOVL # X1007FC,R0  ; Generate Cache Tag Store Addr. 1
          51   00EFF7FC 8F   D0   1E4F  2488  MOVL # XEFF7FC,R1  ; Generate Cache Tag Store Addr. 2
                      60   94   1E56  2489 100$: CLRB (R0)    ; Read Tag Store Address 1 (Tag=100)
                   53   27   DB   1E58  2490  MFPR #CAER,R3   ; Store hit/miss information
                1F 53   00   E0   1E5B  2491  BBS #0,R3,120$   ; If Cache hit, continue
          14 0000FE00 EF   1F   E1   1E5F  2492  BBC #31,APT_CTRL_FLAGS,110$ ; Branch if not APT mode
```

```
          0000FE44'EF   7B'AF    3E  1E67  2493  MOVAW B 110$,APT_ERR_PC ; Save Error PC
             0000FE40'EF    01    88  1E6F  2494  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         EE09    30  1E76  2495  BSBW APT_COMMAND
                           03    11  1E79  2496  BRB 120$
                                 00  1E7B  2497 110$: HALT    ; ERROR, did not get Cache hit
                                     1E7C  2498       ; R0 contains referenced address
                           D8    11  1E7C  2499  BRB 100$   ; Loop on error
                           61    D4  1E7E  2500 120$: CLRL (R1)   ; Write EFF into Cache Tag Store
                       50  04    C2  1E80  2501  SUBL2 #4,R0   ; Generate next Index Val for Addr 1
                       51  04    C2  1E83  2502  SUBL2 #4,R1   ; Generate next Index Val for Addr 2
          C8 54   000001FF 8F    F3  1E86  2503  AOBLEQ # X1FF,R4,100$  ; Continue for 512 Index locations
                                     1E8E  2504
                                     1E8E  2505 ;
                                     1E8E  2506 ; The next sections of code sequences from the largest Index Value
                                     1E8E  2507 ; of the first half of the Tag Store to the smallest (7FC-000),
                                     1E8E  2508 ; each time reading and checking each Cache Tag Store location for
                                     1E8E  2509 ; a hit (tag=EFF), and then writing the complement into that same
                                     1E8E  2510 ; location (tag=100).
                                     1E8E  2511 ;
                                     1E8E  2512
                           54    D4  1E8E  2513  CLRL R4   ; Initialize Index Counter
             50    001007FC 8F    D0  1E90  2514  MOVL # X1007FC,R0  ; Generate Cache Tag Store Addr. 1
             51    00EFF7FC 8F    D0  1E97  2515  MOVL # XEFF7FC,R1  ; Generate Cache Tag Store Addr. 2
                           61    94  1E9E  2516 130$: CLRB (R1)   ; Read Tag Store Address 2 (tag=EFF)
                       53  27    DB  1EA0  2517  MFPR #CAER,R3  ; Store hit/miss information
                     1F 53  00    E0  1EA3  2518  BBS #0,R3,150$  ; If Cache hit, continue test
          14 0000FE00'EF   1F    E1  1EA7  2519  BBC #31,APT_CTRL_FLAGS,140$ ; Branch if not APT mode
          0000FE44 EF   C3'AF    3E  1EAF  2520  MOVAW B 140$,APT_ERR_PC ; Save Error PC
             0000FE40'EF    01    88  1EB7  2521  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         EDC1    30  1EBE  2522  BSBW APT_COMMAND
                           03    11  1EC1  2523  BRB 150$
                                 00  1EC3  2524 140$: HALT    ; ERROR, did not get Cache hit
                                     1EC4  2525       ; R1 contains referenced address
                           D8    11  1EC4  2526  BRB 130$   ; Loop on error
                           60    D4  1EC6  2527 150$: CLRL (R0)   ; Write 100 into Cache Tag Store
                       50  04    C2  1EC8  2528  SUBL2 #4,R0   ; Generate next Index Val for Addr. 1
                       51  04    C2  1ECB  2529  SUBL2 #4,R1   ; Generate next Index Val for Addr. 2
          C8 54   000001FF 8F    F3  1ECE  2530  AOBLEQ # X1FF,R4,130$  ; Continue for 512 Index locations.
                         0127    31  1ED6  2531  BRW T9_S3   ; Go to next subtest.
                          00001ED9  1ED9  2532  ENDTST = .
                          FFFFFFC5  1ED9  2533  .IF GE<ENDTST-BGNTST- X1A0> ; Check test length
                                    1ED9  2534  .ERROR ; MAXIMUM TEST LENGTH OF 1A0 HEX BYTES EXCEEDED!
                                    1ED9  2535  .ENDC    ; End conditional
                                    1ED9  2536
                                    1ED9  2537 ;
                                    1ED9  2538 ; The following code fills the second half of the 1024 Cache Tag Store
                                    1ED9  2539 ; locations (800-FFC), with 100 Hex.
                                    1ED9  2540 ;
                                    1ED9  2541
                          00000ED9  1ED9  2542  TEMP = . & XFFF  ; Put Index value into TEMP
                          000008D9  1ED9  2543  .IF LE <TEMP- X600>  ; If Index LE 600, do not change
                                    1ED9  2544  .IFF    ; If false, move Index up to 000
                          00002000  1ED9  2545  . = <. & XFFFFF000 + X1000> ; Bump up Index to 000 (adds 1000).
                                    2000  2546  .ENDC    ; End conditional
                          00002000  2000  2547  BGNTST = .
             EA01 CF    FD AF    DE  2000  2548 T9_S3: MOVAL T9_S3,RET_PC  ; Store Return PC
                       25  00    DA  2006  2549  MTPR #0,#CADR  ; Flush the Cache
```

```
                    27   0F    DA  2009  2550  MTPR # XF,#CAER  ; Clear the CAER
                         54    D4  200C  2551  CLRL R4   ; Initialize Index Counter
          50   00100800 8F    D0  200E  2552  MOVL # X100800,R0  ; Generate Cache Tag Store Addr. 1
                         80    D4  2015  2553 10$: CLRL (R0)+   ; Write 100 into Cache Tag Store
       F6 54   000001FF 8F    F3  2017  2554  AOBLEQ # X1FF,R4,10$  ; Continue for 512 Index locations
                                  201F  2555
                                  201F  2556
                                  201F  2557
                                  201F  2558 ;
                                  201F  2559 ; The next section of code reads and checks each Tag Store location
                                  201F  2560 ; for a hit (tag=100), and then writes the complement into that same
                                  201F  2561 ; location (tag=EFF). This continues for incrementing through the last
                                  201F  2562 ; 512 locations.
                                  201F  2563 ;
                                  201F  2564
                         54    D4  201F  2565  CLRL R4   ; Initialize Index Counter
          50   00100800 8F    D0  2021  2566  MOVL # X100800,R0  ; Generate Cache Tag Store Addr. 1
          51   00EFF800 8F    D0  2028  2567  MOVL # XEFF800,R1  ; Generate Cache Tag Store Addr. 2
                         60    94  202F  2568 20$: CLRB (R0)   ; Read Tag Store Address 1 (100)
                      53   27    DB  2031  2569  MFPR #CAER,R3  ; Store hit/miss information
                   1F 53   00    E0  2034  2570  BBS #0,R3,40$  ; If Cache hit, continue test
       14 0000FE00'EF     1F    E1  2038  2571  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
       0000FE44 EF   54'AF    3E  2040  2572  MOVAW B 30$,APT_ERR_PC ; Save Error PC
          0000FE40'EF   01    88  2048  2573  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      EC30    30  204F  2574  BSBW APT_COMMAND
                         03    11  2052  2575  BRB 40$
                         00    00  2054  2576 30$: HALT    ; ERROR, did not get Cache hit
                               2055  2577      ; R0 contains referenced address
                         D8    11  2055  2578  BRB 20$   ; Loop on error
                         61    D4  2057  2579 40$: CLRL (R1)   ; Write EFF into Cache Tag Store
                      50   04    C0  2059  2580  ADDL2 #4,R0   ; Generate next Index value for addr 1
                      51   04    C0  205C  2581  ADDL2 #4,R1   ; Generate next Index value for addr 2
       C8 54   000001FF 8F    F3  205F  2582  AOBLEQ # X1FF,R4,20$  ; Continue for 512 Index locations
                                  2067  2583
                                  2067  2584 ;
                                  2067  2585 ; The next section of code reads and checks each Cache Tag Store
                                  2067  2586 ; location for a hit (tag=EFF), and then writes the complement into
                                  2067  2587 ; that same location (tag=100). This continues for incrementing through
                                  2067  2588 ; the last 512 Index locations.
                                  2067  2589 ;
                                  2067  2590
                                  2067  2591
                         54    D4  2067  2592  CLRL R4   ; Initialize Index Counter
          50   00100800 8F    D0  2069  2593  MOVL # X100800,R0  ; Generate Cache Tag Store Addr. 1
          51   00EFF800 8F    D0  2070  2594  MOVL # XEFF800,R1  ; Generate Cache Tag Store Addr. 2
                         61    94  2077  2595 50$: CLRB (R1)   ; Read Tag Store Address 2 (EFF)
                      53   27    DB  2079  2596  MFPR #CAER,R3  ; Store hit/miss information
                   1F 53   00    E0  207C  2597  BBS #0,R3,70$  ; If Cache hit, continue test
       14 0000FE00'EF     1F    E1  2080  2598  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
       0000FE44 EF   9C AF    3E  2088  2599  MOVAW B 60$,APT_ERR_PC ; Save Error Pc
          0000FE40'EF   01    88  2090  2600  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      EBE8    30  2097  2601  BSBW APT_COMMAND
                         03    11  209A  2602  BRB 70$
                               209C  2603 60$: HALT    ; ERROR, did not get Cache hit
                               209D  2604      ; R1 contains referenced address
                         D8    11  209D  2605  BRB 50$   ; Loop on error
                         60    D4  209F  2606 70$: CLRL (R0)   ; Write 100 into Cache Tag Store
```

```
                50    04    C0   20A1   2607  ADDL2 #4,R0    ; Generate next Index value for addr 1
                51    04    C0   20A4   2608  ADDL2 #4,R1    ; Generate next Index value for addr 2
        C8 54   000001FF 8F   F3   20A7   2609  AOBLEQ #X1FF,R4,50$  ; Continue for 512 Index locations
                      19    11   20AF   2610  BRB T9_S4    ; Go to next subtest
                                   20B1   2611
                                   20B1   2612 ;
                                   20B1   2613 ; The next section of code is used only to initialize conditions to
                                   20B1   2614 ; execute T9_S2 in event a scope loop is desired as a result of a
                                   20B1   2615 ; Machine Check Exception. This code shortens the scope loop.
                                   20B1   2616 ;
                                   20B1   2617
                                   20B1   2618 T9_INIT2:     ; Return label for T9_S4 loop on error
                25    00    DA   20B1   2619  MTPR #0,#CADR   ; Flush the Cache
                27    0F    DA   20B4   2620  MTPR #XF,#CAER   ; Clear the CAER
                      54    D4   20B7   2621  CLRL R4    ; Initialize Index Counter
           50   00100000 8F   D0   20B9   2622  MOVL #X100000,R0   ; Generate Cache Tag Store Addr. 1
                      80    D4   20C0   2623 10$: CLRL (R0)+    ; Write 100 into Cache Tag Store
        F6 54   000001FF 8F   F3   20C2   2624  AOBLEQ #X1FF,R4,10$  ; Continue for 512 Index locations
                                   20CA   2625
                                   20CA   2626
                                   20CA   2627 ;
                                   20CA   2628 ; The next section of code sequences from the largest Index value
                                   20CA   2629 ; of the second half of the Tag Store to the smallest(FFC-800), each time
                                   20CA   2630 ; reading and checking each Cache Tag Store location for a hit (tag=100),
                                   20CA   2631 ; and then writing the complement into that same location (tag=EFF).
                                   20CA   2632 ;
                                   20CA   2633
         E937 CF    E4 AF    DE   20CA   2634 T9_S4: MOVAL T9_INIT2,RET_PC   ; Store RET PC for T9_S4 loop on err.
                      54    D4   20D0   2635  CLRL R4    ; Initialize Index Counter
           50   00100FFC 8F   D0   20D2   2636  MOVL #X100FFC,R0   ; Generate Cache Tag Store Addr. 1
           51   00EFFFFC 8F   D0   20D9   2637  MOVL #XEFFFFC,R1   ; Generate Cache Tag Store Addr. 2
                      60    94   20E0   2638 100$: CLRB (R0)    ; Read Tag Store Address 1 (Tag=100)
                   53    27    DB   20E2   2639  MFPR #CAER,R3   ; Store hit/miss information
                 1F 53    00    E0   20E5   2640  BBS #0,R3,120$   ; If Cache hit, continue
        14 0000FE00'EF    1F    E1   20E9   2641  BBC #31,APT_CTRL_FLAGS,110$ ; Branch if not APT mode
        0000FE44'EF    05'AF    3E   20F1   2642  MOVAW B 110$,APT_ERR_PC ; Save Error PC
           0000FE40'EF    01    88   20F9   2643  BISB2 #1,APT_ERR_FLAG   ; Set Error Flag
                      EB7F    30   2100   2644  BSBW APT_COMMAND
                         03    11   2103   2645  BRB 120$
                               00   2105   2646 110$: HALT     ; ERROR, did not get Cache hit
                                   2106   2647      ; R0 contains referenced address
                         D8    11   2106   2648  BRB 100$    ; Loop on error
                         61    D4   2108   2649 120$: CLRL (R1)    ; Write EFF into Cache Tag Store
                   50    04    C2   210A   2650  SUBL2 #4,R0    ; Generate next Index Val for Addr 1
                   51    04    C2   210D   2651  SUBL2 #4,R1    ; Generate next Index Val for Addr 2
        C8 54   000001FF 8F   F3   2110   2652  AOBLEQ #X1FF,R4,100$  ; Continue for 512 Index locations
                                   2118   2653
                                   2118   2654 ;
                                   2118   2655 ; The next sections of code sequences from the largest Index Value
                                   2118   2656 ; of the second half of the Tag Store to the smallest (FFC-800),
                                   2118   2657 ; each time reading and checking each Cache Tag Store location for
                                   2118   2658 ; a hit (tag=EFF), and then writing the complement into that same
                                   2118   2659 ; location (tag=100).
                                   2118   2660 ;
                                   2118   2661
                      54    D4   2118   2662  CLRL R4    ; Initialize Index Counter
           50   00100FFC 8F   D0   211A   2663  MOVL #X100FFC,R0   ; Generate Cache Tag Store Addr. 1
```

```
                51    00EFFFFC 8F    D0  2121  2664  MOVL #^XEFFFFC,R1  ; Generate Cache Tag Store Addr. 2
                                61    94  2128  2665 130$: CLRB (R1)   ; Read Tag Store Address 2 (tag=EFF)
                            53  27    DB  212A  2666  MFPR #CAER,R3  ; Store hit/miss information
                         1F 53  00    E0  212D  2667  BBS #0,R3,150$  ; If Cache hit, continue test
            14 0000FE00'EF     1F    E1  2131  2668  BBC #31,APT_CTRL_FLAGS,140$ ; Branch if not APT mode
            0000FE44'EF     4D AF    3E  2139  2669  MOVAW B 140$,APT_ERR_PC ; Save Error PC
               0000FE40 EF     01    88  2141  2670  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                             EB37    30  2148  2671  BSBW APT_COMMAND
                                03    11  214B  2672  BRB 150$
                                00    12  214D  2673 140$: HALT    ; ERROR, did not get Cache hit
                                      214E  2674      ; R1 contains referenced address
                                D8    11  214E  2675  BRB 130$   ; Loop on error
                                60    D4  2150  2676 150$: CLRL (R0)   ; Write 100 into Cache Tag Store
                            50  04    C2  2152  2677  SUBL2 #4,R0   ; Generate next Index Val for Addr. 1
                            51  04    C2  2155  2678  SUBL2 #4,R1   ; Generate next Index Val for Addr. 2
            C8 54   000001FF 8F    F3  2158  2679  AOBLEQ #^X1FF,R4,130$  ; Continue for 512 Index locations.
                  E89F CF     01    88  2160  2680  BISB2 #1,NXM_WT_FLAG  ; Set flag for pending Wt. NXM Intr.
                            12  00    DA  2165  2681  MTPR #0,#IPL   ; Allow interrupt to occur
                                00002168  2168  2682  ENDTST = .
                                FFFFFFD8  2168  2683  .IF GE<ENDTST-BGNTST- ^X190> ; Check test length
                                          2168  2684  .ERROR ; MAXIMUM TEST LENGTH OF 190 HEX BYTES EXCEEDED!
                                          2168  2685  .ENDC    ; End conditional
```

```
                        2168  2687  .SBTTL  TEST 10:  CACHE DATA STORE DYNAMIC MEMORY TEST
                        2168  2688
                        2168  2689 ;++
                        2168  2690 ; FUNCTIONAL DESCRIPTION:
                        2168  2691 ;
                        2168  2692 ; This test verifies the ability to write and read all of the
                        2168  2693 ; locations in the Cache Data Store Rams by use of a marching
                        2168  2694 ; pattern applied across CACHE<31:00>.
                        2168  2695 ;
                        2168  2696 ; The routine initially writes all the Cache Data Store
                        2168  2697 ; locations with zeroes (00000000). The first Index location
                        2168  2698 ; is then read and checked for correct data, and its complement
                        2168  2699 ; (FFFFFFFF) is then written into that location. This type
                        2168  2700 ; of sequence increments and decrements throughout all of the
                        2168  2701 ; Data Store locations in the Cache.
                        2168  2702 ;
                        2168  2703 ; The starting Physical Address used to reference the Index
                        2168  2704 ; locations is 010000.
                        2168  2705 ;
                        2168  2706 ; The test is divided into 4 subtests:
                        2168  2707 ;
                        2168  2708 ; Subtest 1 checks the first half of the Data Store locations,
                        2168  2709 ; (Index 000-7FC) while incrementing through the Index.
                        2168  2710 ;
                        2168  2711 ; Subtest 2 checks the first half of the Data Store locations,
                        2168  2712 ; (Index 7FC-000) while decrementing through the Index.
                        2168  2713 ;
                        2168  2714 ; Subtest 3 checks the second half of the Data Store locations,
                        2168  2715 ; (Index 800-FFC) while incrementing through the Index.
                        2168  2716 ;
                        2168  2717 ; Subtest 4 checks the second half of the Data Store locations,
                        2168  2718 ; (Index FFC-800) while decrementing through the Index.
                        2168  2719 ;
                        2168  2720 ;
                        2168  2721 ;
                        2168  2722 ;--
                        2168  2723
                        2168  2724
                        2168  2725  SEQ  $TN    ; Macro call for Test Sequence Check
                        2168       TEST_10::     ; Continued on next line
     0000FE50'EF    0A   B0  2168        MOVW #10,APT_TEST_NUM ; Load the Test Number
          0000FE52'EF    B6  216F        INCW APT_TEST_COUNT  ; Bump up the test count
          0000FE50 EF    B1  2175        CMPW APT_TEST_NUM,-  ; Continued on next line
          0000FE52 EF        217B         APT_TEST_COUNT  ; Check for the correct test number
                20   13  2180        BEQL T10_S1    ; Sequence correct, continue test
     0000FE00'EF    1F   E1  2182        BBC #31,APT_CTRL_FLAGS,-
                17       2189         T10_HLT    ; Branch if not APT mode
0000FE44 EF    000021A1'EF   9E  218A        MOVAB T10_HLT,APT_ERR_PC ; Save the error PC
     0000FE40'EF    01   B0  2195        MOVW #1,APT_ERR_FLAG  ; Set the error flag
               EAE3   30  219C        BSBW APT_COMMAND  ; Call APT routine
                01   11  219F        BRB T10_S1    ; Continue
                        21A1       T10_HLT:
                00      21A1        HALT    ; ERROR!!! Program out of sequence
                        21A2       T10_S1:
      E85F CF   FD AF   DE  21A2        MOVAL T10_S1.RET_PC  ; Store return PC
                        21A8
                0655   31  21A8  2726  BRW 5$    ; Go to start of test
```

```
                                21AB  2727 ;
                                21AB  2728 ; The following code fills the first half of the 1024 Cache Data Store
                                21AB  2729 ; locations (000-7FC) with 00000000.
                                21AB  2730 ;
                                21AB  2731
                    000001AB    21AB  2732  TEMP = . & ^XFFF  ; Put Index value into TEMP
                    FFFFF9AB    21AB  2733  .IF LT <TEMP- X800>  ; If Index LT 800, make 800
                    00002800    21AB  2734  . = . +< X800-TEMP>  ; Make Index value equal to 800
                                2800  2735  .IFF    ; If false, go to next conditional
                                2800  2736  .IF GT <TEMP- XE00>  ; If Index GT E00, make 800
                                2800  2737  . = <. & ^XFFFFF000 + ^X1800> ; Bump up Index to 800
                                2800  2738  .ENDC    ; End conditional
                                2800  2739  .ENDC    ; End conditional
                    00002800    2800  2740  BGNTST = .
                25    00    DA  2800  2741 5$: MTPR #0,#CADR  ; Flush the Cache
                27    0F    DA  2803  2742  MTPR # ^XF,#CAER  ; Clear the CAER
                      51    D4  2806  2743  CLRL R1    ; Load Data pattern
                      54    D4  2808  2744  CLRL R4    ; Initialize Index Counter
          50    00010000  8F    D0  280A  2745  MOVL # ^X10000,R0  ; Generate Cache Index value
                      80    D4  2811  2746 10$: CLRL (R0)+   ; Write zeroes into Cache Data Store
       F6 54    000001FF  8F    F3  2813  2747  AOBLEQ # ^X1FF,R4,10$  ; Continue for 512 Index locations
                                281B  2748
                                281B  2749 ;
                                281B  2750 ; The next section of code reads and checks each Data Store location
                                281B  2751 ; for a value of 00000000, and then writes FFFFFFFF into that same
                                281B  2752 ; location. This continues for incrementing through the f'rst 512
                                281B  2753 ; locations.
                                281B  2754 ;
                                281B  2755
                      54    D4  281B  2756  CLRL R4    ; Initialize Index Counter
          50    00010000  8F    D0  281D  2757  MOVL # ^X10000,R0  ; Generate first Index location
                      52    60  D0  2824  2758 20$: MOVL (R0),R2    ; Read Cache Data Store
                53    52    51    CD  2827  2759  XORL3 R1,R2,R3  ; Check for correct data results
                      1F    13  282B  2760  BEQL 40$    ; If data correct, continue test
       14 0000FE00  EF    1F    E1  282D  2761  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
       0000FE44'EF     49  AF    3E  2835  2762  MOVAW B 30$,APT_ERR_PC ; Save Error PC
          0000FE40  EF    01    88  283D  2763  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                    E43B    30    2844  2764  BSBW APT_COMMAND
                      03    11  2847  2765  BRB 40$
                      00    2849  2766 30$: HALT    ; ERROR, Data incorrect
                                284A  2767       ; R0 contains referenced address
                                284A  2768       ; R1 contains data written
                                284A  2769       ; R2 contains data read
                                284A  2770       ; R3 contains bad Data Bit positions
                      D8    ''  284A  2771  BRB 20$    ; Loop on error
          80    FFFFFFFF  8F    D0  284C  2772 40$: MOVL # ^XFFFFFFFF,(R0)+ ; Write all ones into Data Store
       C9 54    000001FF  8F    F3  2853  2773  AOBLEQ # ^X1FF,R4,20$  ; Continue for 512 Index locations
                                285B  2774
                                285B  2775 ;
                                285B  2776 ; The next section of code reads and checks each Cache Data Store
                                285B  2777 ; location for a value of FFFFFFFF and then writes 00000000 into
                                285B  2778 ; that same location. This sequence continues for incrementing
                                285B  2779 ; through the first 512 locations.
                                285B  2780 ;
                                285B  2781
          51    FFFFFFFF  8F    D0  285B  2782  MOVL # ^XFFFFFFFF,R1  ; Save previous Data written
                      54    D4  2862  2783  CLRL R4    ; Initialize Index Counter
```

```
                50    00010000 8F    D0  2864  2784  MOVL #X10000,R0  ; Generate first Index location
                           52  60    D0  286B  2785 50$: MOVL (R0),R2   ; Read Cache Data Store
                       53  52  51    CD  286E  2786  XORL3 R1,R2,R3  ; Check for correct data results
                               1F    13  2872  2787  BEQL 70$   ; If data correct, continue test
            14 0000FE00'EF     1F    E1  2874  2788  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
            0000FE44 EF   90'AF     3E  287C  2789  MOVAW B 60$,APT_ERR_PC ; Save Error PC
               0000FE40'EF     01    88  2884  2790  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          E3F4   30  288B  2791  BSBW APT_COMMAND
                               03    11  288E  2792  BRB 70$
                               00    2890  2793 60$: HALT    ; ERROR, data incorrect
                                     2891  2794      ; R0 contains referenced address
                                     2891  2795      ; R1 contains data written
                                     2891  2796      ; R2 contains data read
                                     2891  2797      ; R3 contains bad Data Bit positions
                               D8    11  2891  2798  BRB 50$   ; Loop on error
                               80    D4  2893  2799 70$: CLRL (R0)+   ; Write zeroes into Cache Data Store
            CE 54    000001FF 8F    F3  2895  2800  AOBLEQ #X1FF,R4,50$  ; Continue for 512 Index locations
                               1B    11  289D  2801  BRB T10_S2   ; Go to next subtest
                                     289F  2802
                                     289F  2803 ;
                                     289F  2804 ; The next section of code is used only to initialize conditions to
                                     289F  2805 ; execute T10_S2 in event a scope loop is desired as a result of a
                                     289F  2806 ; Machine Check Exception. This code shortens the scope loop.
                                     289F  2807 ;
                                     289F  2808
                                     289F  2809 T10_INIT1:    ; Return label for T10_S2 loop on err.
                           25  00    DA  289F  2810  MTPR #0,#CADR  ; Flush the Cache
                           27  0F    DA  28A2  2811  MTPR #XF,#CAER  ; Clear the CAER
                               51    D4  28A5  2812  CLRL R1   ; Load Data pattern
                               54    D4  28A7  2813  CLRL R4   ; Initialize Index Counter
                50    00010000 8F    D0  28A9  2814  MOVL #X10000,R0  ; Generate Cache Index value
                               80    D4  28B0  2815 10$: CLRL (R0)+   ; Write zeroes into Cache Data Store
            F6 54    000001FF 8F    F3  28B2  2816  AOBLEQ #X1FF,R4,10$  ; Continue for 512 Index locations
                                     28BA  2817
                                     28BA  2818 ;
                                     28BA  2819 ; The next section of code sequences from the largest Index value
                                     28BA  2820 ; of the first half of the Data Store to the smallest (7FC-000),
                                     28BA  2821 ; each time reading and checking each Cache Data Store location
                                     28BA  2822 ; for a value of 00000000, and then writing FFFFFFFF into
                                     28BA  2823 ; that same location.
                                     28BA  2824 ;
                                     28BA  2825
                E147 CF    E2 AF    DE  28BA  2826 T10_S2: MOVAL T10_INIT1,RET_PC ; Store RET PC for T10_S2 loop on err.
                               51    D4  28C0  2827  CLRL R1   ; Save previous Data written
                               54    D4  28C2  2828  CLRL R4   ; Initialize Index Counter
                50    000107FC 8F    D0  28C4  2829  MOVL #X107FC,R0  ; Generate highest Index position
                           52  60    D0  28CB  2830 80$: MOVL (R0),R2   ; Read Cache Data Store
                       53  52  51    CD  28CE  2831  XORL3 R1,R2,R3  ; Check for correct data results
                               1F    13  28D2  2832  BEQL 100$   ; If data correct, continue test
            14 0000FE00 EF     1F    E1  28D4  2833  BBC #31,APT_CTRL_FLAGS,90$ ; Branch if not APT mode
            0000FE44 EF   F0'AF     3E  28DC  2834  MOVAW B 90$,APT_ERR_PC ; Save Error PC
               0000FE40'EF     01    88  28E4  2835  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          E394   30  28EB  2836  BSBW APT_COMMAND
                               03    11  28EE  2837  BRB 100$
                               00    28F0  2838 90$: HALT    ; ERROR, Data incorrect
                                     28F1  2839      ; R0 contains referenced address
                                     28F1  2840      ; R1 contains data written
```

```
                                28F1  2841       ; R2 contains data read
                                28F1  2842       ; R3 contains bad Data Bit positions
                          D8    11    28F1  2843  BRB 80$    ; Loop on error
            60    FFFFFFFF  8F    D0    28F3  2844 100$: MOVL #^XFFFFFFFF,(R0) ; Write all ones into Cache Data Store
                      50    04    C2    28FA  2845  SUBL2 #4,R0    ; Generate next Index location
         C6 54    000001FF  8F    F3    28FD  2846  AOBLEQ #^X1FF,R4,80$  ; Continue for 512 Index locations
                                2905  2847
                                2905  2848 ;
                                2905  2849 ; The next section of code sequences from the largest Index value
                                2905  2850 ; of the first half of the Data Store to the smallest (7FC-000),
                                2905  2851 ; each time reading and checking each Cache Data Store location
                                2905  2852 ; for a value of FFFFFFFF and then writing 00000000 into
                                2905  2853 ; that same location
                                2905  2854 ;
                                2905  2855
            51    FFFFFFFF  8F    D0    2905  2856  MOVL #^XFFFFFFFF,R1   ; Save previous data written
                            54    D4    290C  2857  CLRL R4    ; Initialize Index Counter
            50    000107FC  8F    D0    290E  2858  MOVL #^X107FC,R0   ; Generate highest Index location
                      52    60    D0    2915  2859 110$: MOVL (R0),R2    ; Read Cache Data Store
                  53  52    51    CD    2918  2860  XORL3 R1,R2,R3   ; Check for correct data results
                            1F    13    291C  2861  BEQL 130$    ; If data correct, continue test
         14 0000FE00  EF    1F    E1    291E  2862  BBC #31,APT_CTRL_FLAGS,120$ ; Branch if not APT mode
         C000FE44'EF     3A'AF    3E    2926  2863  MOVAW B 120$,APT_ERR_PC ; Save Error PC
            0000FE40  EF    01    88    292E  2864  BISB2 #1,APT_ERR_FLAG   ; Set Error Flag
                        E34A    30    2935  2865  BSBW APT_COMMAND
                            03    11    2938  2866  BRB 130$
                                00    293A  2867 120$: HALT     ; ERROR, data incorrect
                                      293B  2868       ; R0 contains referenced address
                                      293B  2869       ; R1 contains data written
                                      293B  2870       ; R2 contains data read
                                      293B  2871       ; R3 contains bad Data Bit positions
                          D8    11    293B  2872  BRB 110$    ; Loop on error
                          60    D4    293D  2873 130$: CLRL (R0)    ; Write zeroes into Cache Data Store
                      50    04    C2    293F  2874  SUBL2 #4,R0    ; Generate next Index value
         CB 54    000001FF  8F    F3    2942  2875  AOBLEQ #^X1FF,R4,110$  ; Continue for 512 Index locations
                        06B3    31    294A  2876  BRW T10_S3    ; Go to next subtest
                         0000294D       294D  2877  ENDTST = .
                         FFFFFFAD       294D  2878  .IF GE<ENDTST-BGNTST- X1A0> ; Check test length
                                      294D  2879  .ERROR ; MAXIMUM TEST LENGTH OF 1A0 HEX BYTES EXCEEDED!
                                      294D  2880  .ENDC     ; End conditional
                                      294D  2881
                                      294D  2882 ;
                                      294D  2883 ; The following code fills the second half of the 1024 Cache Data Store
                                      294D  2884 ; locations (800-FFC) with 00000000.
                                      294D  2885 ;
                                      294D  2886
                         0000094D     294D  2887  TEMP = . & ^XFFF   ; Put Index value into TEMP
                         0000034D     294D  2888  .IF LE <TEMP- X600>  ; If Index LE 600, do not change
                                      294D  2889  .IFF     ; If false, make Index value 000
                         00003000     294D  2890  . = <. & ^XFFFFF000 + ^X1000> ; Bump up Index to 000 (add 1000)
                                      3000  2891  .ENDC     ; End conditional
                         00003000     3000  2892  BGNTST = .
            DA01 CF    FD AF    DE    3000  2893 T10_S3: MOVAL T10_S3,RET_PC  ; Store RET PC for T10_S3 loop on err.
                    25    00    DA    3006  2894  MTPR #0,#CADR   ; Flush the Cache
                    27    0F    DA    3009  2895  MTPR #^XF,#CAER   ; Clear the CAER
                          51    D4    300C  2896  CLRL R1    ; Load Data pattern
                          54    D4    300E  2897  CLRL R4    ; Initialize Index Counter
```

```
                50    00010800 8F    D0  3010  2898  MOVL # X10800,R0  ; Generate Cache Index value
                                80    D4  3017  2899 10$: CLRL (R0)+   ; Write zeroes into Cache Data Store
              F6 54    000001FF 8F    F3  3019  2900  AOBLEQ # X1FF,R4,10$  ; Co tinue for 512 Index locations
                                      3021  2901
                                      3021  2902 ;
                                      3021  2903 ; The next section of code reads and checks each Data Store location
                                      3021  2904 ; for a value of 00000000, and then writes FFFFFFFF into that same
                                      3021  2905 ; location. This continues for incrementing through the last 512
                                      3021  2906 ; locations.
                                      3021  2907 ;
                                      3021  2908
                                54    D4  3021  2909  CLRL R4   ; Initialize Index Counter
                50    00010800 8F    D0  3023  2910  MOVL # X10800,R0  ; Generate first Index location
                          52    60    D0  302A  2911 20$: MOVL (R0),R2   ; Read Cache Data Store
                      53    52    51    CD  302D  2912  XORL3 R1,R2,R3  ; Check for correct data results
                                1F    13  3031  2913  BEQL 40$   ; If data correct, continue test
              14 0000FE00'EF    1F    E1  3033  2914  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
              0000FE44 EF    4F'AF    3E  303B  2915  MOVAW B 30$,APT_ERR_PC ; Save Error PC
                 0000FE40'EF    01    88  3043  2916  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                             DC35    30  304A  2917  BSBW APT_COMMAND
                                03    11  304D  2918  BRB 40$
                                00    00  304F  2919 30$: HALT    ; ERROR, Data incorrect
                                      3050  2920      ; R0 contains referenced address
                                      3050  2921      ; R1 contains data written
                                      3050  2922      ; R2 contains data read
                                      3050  2923      ; R3 contains bad Data Bit positions
                                D8    11  3050  2924  BRB 20$   ; Loop on error
                50    FFFFFFFF 8F    D0  3052  2925 40$: MOVL # XFFFFFFFF,(R0)+ ; Write all ones into Data Store
              C9 54    000001FF 8F    F3  3059  2926  AOBLEQ # X1FF,R4,20$  ; Continue for 512 Index locations
                                      3061  2927
                                      3061  2928 ;
                                      3061  2929 ; The next section of code reads and checks each Cache Data Store
                                      3061  2930 ; location for a value of FFFFFFFF and then writes 00000000 into
                                      3061  2931 ; that same location. This sequence continues for incrementing
                                      3061  2932 ; through the last 512 locations.
                                      3061  2933 ;
                                      3061  2934
                51    FFFFFFFF 8F    D0  3061  2935  MOVL # XFFFFFFFF,R1  ; Save previous Data written
                                54    D4  3068  2936  CLRL R4   ; Initialize Index Counter
                50    00010800 8F    D0  306A  2937  MOVL # X10800,R0  ; Generate first Index location
                          52    60    D0  3071  2938 50$: MOVL (R0),R2   ; Read Cache Data Store
                      53    52    51    CD  3074  2939  XORL3 R1,R2,R3  ; Check for correct data results
                                1F    13  3078  2940  BEQL 70$   ; If data correct, continue test
              14 0000FE00 EF    1F    E1  307A  2941  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
              0000FE44 EF    96 AF    3E  3082  2942  MOVAW B 60$,APT_ERR_PC ; Save Error PC
                 0000FE40 EF    01    88  308A  2943  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                             DBEE    30  3091  2944  BSBW APT_COMMAND
                                03    11  3094  2945  BRB 70$
                                00    00  3096  2946 60$: HALT    ; ERROR, data incorrect
                                      3097  2947      ; R0 contains referenced address
                                      3097  2948      ; R1 contains data written
                                      3097  2949      ; R2 contains data read
                                      3097  2950      ; R3 contains bad Data Bit positions
                                D8    11  3097  2951  BRB 50$   ; Loop on error
                                80    D4  3099  2952 70$: CLRL (R0)+   ; Write zeroes into Cache Data Store
              CE 54    000001FF 8F    F3  309B  2953  AOBLEQ # X1FF,R4,50$  ; Continue for 512 Index locations
                                1B    11  30A3  2954  BRB T10_S4   ; Go to next subtest
```

```
                                30A5  2955
                                30A5  2956 ;
                                30A5  2957 ; The next section of code is used only to initialize conditions to
                                30A5  2958 ; execute T10_S4 in event a scope loop is desired as a result of a
                                30A5  2959 ; Machine Check Exception. This code shortens the scope loop.
                                30A5  2960 ;
                                30A5  2961
                                30A5  2962 T10_INIT2:    ; Return label for T10_S4 loop on err.
                 25   00   DA   30A5  2963  MTPR #0,#CADR  ; Flush the Cache
                 27   0F   DA   30A8  2964  MTPR # XF,#CAER  ; Clear the CAER
                      51   D4   30AB  2965  CLRL R1    ; Load Data pattern
                      54   D4   30AD  2966  CLRL R4    ; Initialize Index Counter
           50    00010800 8F   D0   30AF  2967  MOVL # X10800,R0  ; Generate Cache Index value
                      80   D4   30B6  2968 10$: CLRL (R0)+   ; Write zeroes into Cache Data Store
        F6 54    000001FF 8F   F3   30B8  2969  AOBLEQ # X1FF,R4,10$  ; Continue for 512 Index locations
                                30C0  2970
                                30C0  2971
                                30C0  2972 ;
                                30C0  2973 ; The next section of code sequences from the largest Index value
                                30C0  2974 ; of the second half of the Data Store to the smallest (FFC-800),
                                30C0  2975 ; each time reading and checking each Cache Data Store location
                                30C0  2976 ; for a value of 00000000, and then writing FFFFFFFF into
                                30C0  2977 ; that same location.
                                30C0  2978 ;
                                30C0  2979
           D941 CF    E2 AF   DE   30C0  2980 T10_S4: MOVAL T10_INIT2,RET_PC ; Store RET PC for T10_S4 loop on err.
                      51   D4   30C6  2981  CLRL R1    ; Save previous Data written
                      54   D4   30C8  2982  CLRL R4    ; Initialize Index Counter
              50    00010FFC 8F   D0   30CA  2983  MOVL # X10FFC,R0  ; Generate highest Index position
                         52   60   D0   30D1  2984 80$: MOVL (R0),R2   ; Read Cache Data Store
                   53    52   51   CD   30D4  2985  XORL3 R1,R2,R3  ; Check for correct data results
                      1F   13   30D8  2986  BEQL 100$   ; If data correct, continue test
        14 0000FE00 EF    1F   E1   30DA  2987  BBC #31,APT_CTRL_FLAGS,90$ ; Branch if not APT mode
        0000FE44 EF    F6 AF   3E   30E2  2988  MOVAW B 90$,APT_ERR_PC ; Save Error PC
           0000FE40 EF    01   88   30EA  2989  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         DB8E   30   30F1  2990  BSBW APT_COMMAND
                      03   11   30F4  2991  BRB 100$
                           00   30F6  2992 90$: HALT     ; ERROR, Data incorrect
                                30F7  2993       ; R0 contains referenced address
                                30F7  2994       ; R1 contains data written
                                30F7  2995       ; R2 contains data read
                                30F7  2996       ; R3 contains bad Data Bit positions
                      D8   11   30F7  2997  BRB 80$    ; Loop on error
              60    FFFFFFFF 8F   D0   30F9  2998 100$: MOVL # XFFFFFFFF,(R0) ; Write all ones into Cache Data Store
                         50   04   C2   3100  2999  SUBL2 #4,R0    ; Generate next Index location
        C6 54    000001FF 8F   F3   3103  3000  AOBLEQ # X1FF,R4,80$  ; Continue for 512 Index locations
                                310B  3001
                                310B  3002 ;
                                310B  3003 ; The next section of code sequences from the largest Index value
                                310B  3004 ; of the second half of the Data Store to the smallest (FFC-800),
                                310B  3005 ; each time reading and checking each Cache Data Store location
                                310B  3006 ; for a value of FFFFFFFF and then writing 00000000 into
                                310B  3007 ; that same location
                                310B  3008 ;
                                310B  3009
              51    FFFFFFFF 8F   D0   310B  3010  MOVL # XFFFFFFFF,R1  ; Save previous data written
                      54   D4   3112  3011  CLRL R4    ; Initialize Index Counter
```

```
                50    00010FFC 8F    D0  3114  3012  MOVL #X10FFC,R0  ; Generate highest Index location
                            52   60    D0  311B  3013 110$: MOVL (R0),R2   ; Read Cache Data Store
                        53  52   51    CD  311E  3014  XORL3 R1,R2,R3  ; Check for correct data results
                                1F    13  3122  3015  BEQL 130$   ; If data correct, continue test
            14 0000FE00'EF      1F    E1  3124  3016  BBC #31,APT_CTRL_FLAGS,120$ ; Branch if not APT mode
            0000FE44'EF   40'AF       3E  312C  3017  MOVAW B 120$,APT_ERR_PC ; Save Error PC
               0000FE40'EF    01    88  3134  3018  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          DB44    30  313B  3019  BSBW APT_COMMAND
                            03    11  313E  3020  BRB 130$
                                  00  3140  3021 120$: HALT    ; ERROR, data incorrect
                                      3141  3022      ; R0 contains referenced address
                                      3141  3023      ; R1 contains data written
                                      3141  3024      ; R2 contains data read
                                      3141  3025      ; R3 contains bad Data Bit positions
                             D8   11  3141  3026  BRB 110$   ; Loop on error
                             60   D4  3143  3027 130$: CLRL (R0)   ; Write zeroes into Cache Data Store
                        50   04   C2  3145  3028  SUBL2 #4,R0   ; Generate next Index value
            CB 54   000001FF 8F   F3  3148  3029  AOBLEQ #X1FF,R4,110$  ; Continue for 512 Index locations
                          00003150  3150  3030  ENDTST = .
                          FFFFFFC0  3150  3031  .IF GE<ENDTST-BGNTST- X190> ; Check test length
                                    3150  3032  .ERROR ; MAXIMUM TEST LENGTH OF 190 HEX BYTES EXCEEDED!
                                    3150  3033  .ENDC    ; End conditional
```

```
                                3150  3035  .SBTTL  TEST 11:  BYTE WRITE TEST
                                3150  3036
                                3150  3037 ;++
                                3150  3038 ; FUNCTIONAL DESCRIPTION:
                                3150  3039 ;
                                3150  3040 ; This test will verify that one unique byte of a block can
                                3150  3041 ; be written, while not affecting the other three bytes.
                                3150  3042 ;
                                3150  3043 ; A longword of A's is first written into a test location
                                3150  3044 ; (10000), and then a byte of ones (FF) is written into
                                3150  3045 ; byte zero of that test location. The byte write is
                                3150  3046 ; checked for a hit. A read is then performed on the
                                3150  3047 ; original test location (10000), and the data is
                                3150  3048 ; checked for the correct pattern = AAAAAAFF. This method
                                3150  3049 ; is used for writing to all four bytes of the test location.
                                3150  3050 ;
                                3150  3051 ; The four Enable Byte lines to the Data Store Rams are
                                3150  3052 ; checked here.
                                3150  3053 ;
                                3150  3054 ; The four expected patterns are as follows:
                                3150  3055 ;
                                3150  3056 ;  AAAAAAFF
                                3150  3057 ;  AAAAFFAA
                                3150  3058 ;  AAFFAAAA
                                3150  3059 ;  FFAAAAAA
                                3150  3060 ;
                                3150  3061 ;--
                                3150  3062
                                3150  3063  SEQ  $TN    ; Macro call for Test Sequence Check
                                3150        TEST_11::     ; Continued on next line
          0000FE50'EF    0B    B0  3150         MOVW #11,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  3157         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50 EF    B1  315D         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52 EF        3163          APT_TEST_COUNT  ; Check for the correct test number
                        20    13  3168         BEQL T11_S1    ; Sequence correct, continue test
          0000FE00 EF    1F    E1  316A         BBC #31,APT_CTRL_FLAGS,-
                        17        3171          T11_HLT    ; Branch if not APT mode
0000FE44 EF    00003189 EF    9E  3172         MOVAB T11_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01    B0  317D         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                      DAFB    30  3184         BSBW APT_COMMAND  ; Call APT routine
                        01    11  3187         BRB T11_S1    ; Continue
                            3189        T11_HLT:
                        00      3189         HALT     ; ERROR!!! Program out of sequence
                            318A        T11_S1:
           D877 CF    FD AF   DE  318A         MOVAL T11_S1,RET_PC  ; Store return PC
                            3190
                      0000    31  3190  3064  BRW 5$    ; Go to start of test
                       00000193  3193  3065  TEMP = . & XFFF  ; Put present Index value into TEMP
                       0000018F  3193  3066  .IF LE<TEMP-004>  ; If Index LE 004, make it 004
                                 3193  3067  . = . + <004-TEMP>  ; Make Index equal to 004
                                 3193  3068  .IFF     ; If false, go to next conditional
                        FFFFF293  3193  3069  .IF GT<TEMP- XF00>  ; If Index is GT F00 move up to 004
                                 3193  3070  . = <. & XFFFFF000> + X1004 ; Move Index up to next 004
                                 3193  3071  .ENDC     ; End conditional
                                 3193  3072  .ENDC     ; End conditional
                       00003193  3193  3073  BGNTST = .
                  25    00   DA  3193  3074 5$: MTPR #0,#CADR  ; Flush the Cache
```

```
                27    0F    DA  3196  3075  MTPR # XF,#CAER  ; Clear the CAER
                      54    D4  3199  3076  CLRL R4   ; Initialize Counter
     50    00010000 8F    D0  319B  3077  MOVL # X10000,R0  ; Generate Test location (10000)
                51    50    D0  31A2  3078  MOVL R0,R1   ; Make copy of Test location
     52    AAAAAAFF 8F    D0  31A5  3079  MOVL # XAAAAAAFF,R2  ; Load first data pattern = AAAAAAFF
     60    AAAAAAAA 8F    D0  31AC  3080 10$: MOVL # XAAAAAAAA,(R0) ; Initialize test loc. with patt. = A s
              61    FF 8F    90  31B3  3081  MOVB # XFF,(R1)  ; Write ones into byte address in R1
                55    27    DB  31B7  3082  MFPR #CAER,R5  ; Store hit/miss information
              1F 55    00    E0  31BA  3083  BBS #0,R5,30$  ; If Cache hit, continue test
     14 0000FE00 EF    1F    E1  31BE  3084  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
     0000FE44'EF    DA AF    3E  31C6  3085  MOVAW B 20$,APT_ERR_PC ; Save Error PC
        0000FE40 EF    01    88  31CE  3086  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                    DAAA    30  31D5  3087  BSBW APT_COMMAND
                      03    11  31D8  3088  BRB 30$
                            00  31DA  3089 20$: HALT    ; ERROR, did not get Cache hit
                                31DB  3090      ; R1 contains referenced address
                      CF    11  31DB  3091  BRB 10$   ; Loop on error
                53    60    D0  31DD  3092 30$: MOVL (R0),R3   ; Read Test location
                53    52    D1  31E0  3093  CMPL R2,R3   ; Check for correct data results
                      1F    13  31E3  3094  BEQL 50$   ; If data correct, continue test
     14 0000FE00 EF    1F    E1  31E5  3095  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
     0000FE44 EF    01 AF    3E  31ED  3096  MOVAW B 40$,APT_ERR_PC ; Save Error PC
        0000FE40 EF    01    88  31F5  3097  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                    DA83    30  31FC  3098  BSBW APT_COMMAND
                      03    11  31FF  3099  BRB 50$
                            00  3201  3100 40$: HALT    ; ERROR, data incorrect as a result of
                                3202  3101      ; previous byte write.
                                3202  3102      ; R1 contains byte address
                                3202  3103      ; R2 contains expected results,
                                3202  3104      ; with hex FF indicating correct byte
                                3202  3105      ; location.
                                3202  3106      ; R3 contains data read
              A8    11        3202  3107  BRB 10$   ; Loop on error
              51    D6        3204  3108 50$: INCL R1   ; Generate next byte address
  52    52    08    9C        3206  3109  ROTL #8,R2,R2  ; Generate next data pattern
    9E 54     03    F3        320A  3110  AOBLEQ #3,R4,10$  ; Continue for 4 patterns
           0000320E            320E  3111  ENDTST = .
           FFFFFF8B            320E  3112  .IF GE<ENDTST-BGNTST- XF0> ; Check test length
                               320E  3113  .ERROR ; MAXIMUM TEST LENGTH OF F0 HEX BYTES EXCEEDED!
                               320E  3114  .ENDC    ; End conditional
```

```
                                320E  3116  .SBTTL TEST 12:  CACHE I/O SPACE REFERENCE TEST
                                320E  3117
                                320E  3118 ;++
                                320E  3119 ; FUNCTIONAL DESCRIPTION:
                                320E  3120 ;
                                320E  3121 ; This test will verify that a reference to a register
                                320E  3122 ; in I/O Space will not get cached.
                                320E  3123 ;--
                                320E  3124
                                320E  3125
                                320E  3126  SEQ  $TN    ; Macro call for Test Sequence Check
                                320E        TEST_12::     ; Continued on next line
          0000FE50'EF    0C    B0  320E         MOVW #12,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  3215         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B.  321B         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        3221          APT_TEST_COUNT  ; Check for the correct test number
                         20    13  3226         BEQL T12_S1    ; Sequence correct, continue test
          0000FF00'EF    1F    E1  3228         BBC #31,APT_CTRL_FLAGS,-
                         17        322F          T12_HLT    ; Branch if not APT mode
     0000FE44'EF    00003247'EF    9E  3230         MOVAB T12_HLT,APT_ERR_PC ; Save the error PC
          0000FE40'EF    01    B0  323B         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       DA3D    30  3242         BSBW APT_COMMAND  ; Call APT routine
                         01    11  3245         BRB T12_S1    ; Continue
                             3247        T12_HLT:
                         00    00  3247         HALT     ; ERROR!!! Program out of sequence
                             3248        T12_S1:
      D7B9 CF    FD AF    DE  3248         MOVAL T12_S1,RET_PC  ; Store return PC
                             324E
                      0000    31  324E  3127  BRW 5$    ; Go to start of test
                       00000251  3251  3128  TEMP = . & ^XFFF  ; Put present Index value into TEMP
                       0000024D  3251  3129  .IF LE<TEMP-004>  ; If Index LE 004, make it 004
                                 3251  3130  . = . + <004-TEMP>  ; Make Index value equal to 004
                                 3251  3131  .IFF     ; If false, go to next conditional
                       FFFFF351  3251  3132  .IF GT<TEMP- ^XF00>  ; If Index GT F00, move up to 004
                                 3251  3133  . = <. & ^XFFFF000> + ^X1004 ; Move Index up to next 004
                                 3251  3134  .ENDC     ; End conditional
                                 3251  3135  .ENDC     ; End conditional
                       00003251  3251  3136  BGNTST = .
                   25    00    DA  3251  3137 5$: MTPR #0,#CADR  ; Flush the Cache
                   27    0F    DA  3254  3138  MTPR # ^XF,#CAER  ; Clear the CAER
     50    00010000  8F    D0  3257  3139  MOVL # ^X10000,R0  ; Load Test location
     51    00F20000  8F    D0  325E  3140  MOVL # ^XF20000,R1  ; Load I/O Space Address
     60    AAAAAAAA  8F    D0  3265  3141 10$: MOVL # ^XAAAAAAAA,(R0) ; Write pattern into Test location
                       61    D5  326C  3142  TSTL (R1)    ; Make I/O Space reference (F20000)
                       61    D5  326E  3143  TSTL (R1)    ; Make I/O Space reference again
                   52    27    DB  3270  3144  MFPR #CAER,R2  ; Store hit/miss information
                 1F 52    00    E1  3273  3145  BBC #0,R2,30$  ; If no hit, continue test
 14 0000FE00 EF    1F    E1  3277  3146  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
 0000FE44 EF    93 AF    3E  327F  3147  MOVAW B 20$,APT_ERR_PC ; Save Error PC
     0000FE40 EF    01    88  3287  3148  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                   D9F1    30  328E  3149  BSBW APT_COMMAND
                       03    11  3291  3150  BRB 30$
                             00  3293  3151 20$: HALT     ; ERROR, unexpected Cache hit
                                 3294  3152       ; making I/O Space reference
                                 3294  3153       ; R1 contains I/O Space Address
                                 3294  3154       ; R2 contains contents of CAER
                       CF    11  3294  3155  BRB 10$    ; Loop on error
```

```
                53    60    D0  3296  3156 30$: MOVL (R0),R3    ; Read Test location (10000)
                54    27    DB  3299  3157  MFPR #CAER,R4  ; Store hit/miss information
             1F 54    00    E0  329C  3158  BBS #0,R4,50$  ; If Cache hit, continue test
       14 0000FE00'EF    1F    E1  32A0  3159  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
       0000FE44'EF   BC'AF    3E  32A8  3160  MOVAW B^40$,APT_ERR_PC ; Save Error PC
          0000FE40'EF    01    88  32B0  3161  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      D9C8    30  32B7  3162  BSBW APT_COMMAND
                        03    11  32BA  3163  BRB 50$
                        00    32BC  3164 40$: HALT    ; ERROR, did not get Cache hit
                              32BD  3165      ; reading Test location (10000).
                        A6    11  32BD  3166  BRB 10$   ; Loop on error
          53   AAAAAAAA 8F    D1  32BF  3167 50$: CMPL # ^XAAAAAAAA,R3  ; Check for correct data results
                        20    13  32C6  3168  BEQL TEST_13   ; If data correct, go to next test.
       14 0000FE00'EF    1F    E1  32C8  3169  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
       0000FE44 EF   E4'AF    3E  32D0  3170  MOVAW B^60$,APT_ERR_PC ; Save Error PC
          0000FE40'EF    01    88  32D8  3171  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      D9A0    30  32DF  3172  BSBW APT_COMMAND
                        04    11  32E2  3173  BRB 70$
                        00    32E4  3174 60$: HALT    ; ERROR, data incorrect
                              32E5  3175      ; expected data=AAAAAAAA
                              32E5  3176      ; R3 contains data read
                      FF7D    31  32E5  3177  BRW 10$   ; Loop on error
                              32E8  3178 70$:
                       000032E8  32E8  3179  ENDTST = .
                       FFFFFFA7  32E8  3180  .IF GE<ENDTST-BGNTST- ^XF0> ; Check test length
                              32E8  3181  .ERROR ; MAXIMUM TEST LENGTH OF F0 HEX BYTES EXCEEDED!
                              32E8  3182  .ENDC    ; End conditional
```

```
                                32E8  3184  .SBTTL  TEST 13:  NON CPU MEMORY REFERENCE TEST
                                32E8  3185
                                32E8  3186 ;++
                                32E8  3187 ; FUNCTIONAL DESCRIPTION:
                                32E8  3188 ;
                                32E8  3189 ; This test will verify that all writes on the CMI which do
                                32E8  3190 ; not originate from the CPU will be checked for hits, and
                                32E8  3191 ; those hits will be invalidated.  In the case of a miss,
                                32E8  3192 ; no action will be taken
                                32E8  3193 ;
                                32E8  3194 ; Five Test Addresses are needed to check the CMI<23:00> lines.
                                32E8  3195 ;
                                32E8  3196 ; CA_TST_ADR:: .LONG # XAAAAA8
                                32E8  3197 ;    .LONG # X555554
                                32E8  3198 ;    .LONG # XCCCCCC
                                32E8  3199 ;    .LONG # X3C3C3C
                                32E8  3200 ;    .LONG # X7C03FC
                                32E8  3201 ;
                                32E8  3202 ; A Unibus Exerciser Terminator (UET) module is required for this
                                32E8  3203 ; test in order to perform writes on the CMI that do not originate
                                32E8  3204 ; from the CPU.
                                32E8  3205 ;
                                32E8  3206 ; Conditional Assembly Block Directives are needed in this test
                                32E8  3207 ; to prohibit the five Index locations in the Test Addresses from
                                32E8  3208 ; being referenced by the test code.
                                32E8  3209 ;
                                32E8  3210 ; The IPL is raised to 1F at the beginning of the test to block
                                32E8  3211 ; the expected Non-Existent Memory Interrupts that will occur
                                32E8  3212 ; when writing to the Test Addresses.
                                32E8  3213 ;
                                32E8  3214 ; The test is successful is a Read NXM Machine Check occurs
                                32E8  3215 ; as a result of the UET invalidating the Index location
                                32E8  3216 ; of the Test Address. This test address is initially made
                                32E8  3217 ; a hit in Cache.
                                32E8  3218 ;
                                32E8  3219 ;--
                                32E8  3220
                                32E8  3221
                                32E8  3222  SEQ  $TN    ; Macro call for Test Sequence Check
                                32E8        TEST_13::     ; Continued on next line
          0000FE50 EF    0D    B0  32E8         MOVW #13,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  32EF         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50 EF    B1  32F5         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52 EF        32FB          APT_TEST_COUNT  ; Check for the correct test number
                          20   13  3300         BEQL T13_S1    ; Sequence correct, continue test
          0000FE00 EF    1F    E1  3302         BBC #31,APT_CTRL_FLAGS,-
                          17       3309          T13_HLT    ; Branch if not APT mode
0000FE44 EF    00003321'EF    9E  330A         MOVAB T13_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01    B0  3315         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       D963    30  331C         BSBW APT_COMMAND  ; Call APT routine
                         01    11  331F         BRB T13_S1    ; Continue
                                   3321        T13_HLT:
                               00  3321         HALT    ; ERROR!!! Program out of sequence
                                   3322        T13_S1:
           D6DF CF    FD AF    DE  3322         MOVAL T13_S1,RET_PC  ; Store return PC
                                   3328
                        00D5    31  3328  3223  BRW 5$    ; Go to start of test
```

```
                0000032B  332B  3224  TEMP = . & XFFF  ; Put Index value into TEMP
                0000002B  332B  3225  .IF LT<TEMP- X300>  ; If Index LT 300, do not change
                          332B  3226  .IFF    ; If false, go to next conditional
                FFFFFF2B  332B  3227  .IF LE<TEMP- X400>  ; If Index LE 400, make it 400
                00003400  332B  3228  . = . + < X400-TEMP>  ; Make Index value equal to 400
                          3400  3229  .IFF    ; If false, go to next conditional
                          3400  3230  .IF LT<TEMP- X450>  ; If Index LT 450, do not change
                          3400  3231  .IFF    ; If false, go to next conditional
                          3400  3232  .IF LE<TEMP- X55C>  ; If Index LE 55C, make it 55C
                          3400  3233  . = . + < X55C-TEMP>  ; Make Index value equal to 55C
                          3400  3234  .IFF    ; If false, go to next conditional
                          3400  3235  .IF LT<TEMP- X9A0>  ; If Index LT 9A0, do not change
                          3400  3236  .IFF    ; If false, to to next conditional
                          3400  3237  .IF LE<TEMP- XAAC>  ; If Index LE AAC, make it AAC
                          3400  3238  . = . + < XAAC-TEMP>  ; Make Index value equal to AAC
                          3400  3239  .IFF    ; If false, go to next conditional
                          3400  3240  .IF LT<TEMP- XB40>  ; If Index LT B40, do not change
                          3400  3241  .IFF    ; If false, go to next conditional
                          3400  3242  .IF LE<TEMP- XCD0>  ; If Index LE CD0, make it CD0
                          3400  3243  . = . + < XCD0-TEMP>  ; Make Index value equal to CD0
                          3400  3244  .ENDC    ; End conditional
                          3400  3245  .ENDC    ; End conditional
                          3400  3246  .ENDC    ; End conditional
                          3400  3247  .ENDC    ; End conditional
                          3400  3248  .ENDC    ; End conditional
                          3400  3249  .ENDC    ; End conditional
                          3400  3250  .ENDC    ; End conditional
                          3400  3251  .ENDC    ; End conditional
                    00003400  3400  3252  BGNTST = .
                12    1F    DA  3400  3253 5$: MTPR # X1F,#IPL  ; Raise IPL to block exp. Wt NXM Intr.
                54    01    D0  3403  3254  MOVL #1,R4   ; Initialize counter
                27    0F    DA  3406  3255  MTPR # XF,#CAER  ; Clear the CAER
             50    CEBB CF    DE  3409  3256  MOVAL CA_TST_ADR,R0  ; Set up Test Address pointer
             51    CECA CF    DE  340E  3257  MOVAL UNI_PFN,R1  ; Set up Unibus PFN pcinter
                25    00    DA  3413  3258 10$: MTPR #0,#CADR  ; Flush the Cache
                37    00    DA  3416  3259  MTPR #0,#IO_RESET  ; Perform I/O initialization
00FFF464'EF    8000  8F    A8  3419  3260  BISW2 # X8000,UET_CNTL ; Initialize UET
          D5DD CF    01    88  3422  3261  BISB2 #1,NXM_WT_FLAG  ; Set flag for expected Write to NXM
          D5D9 CF    01    88  3427  3262  BISB2 #1,NXM_RD_FLAG  ; Set flag for expected Reads to NXM
D5D6 CF    00003465'EF    DE  342C  3263  MOVAL 40$,CONT_PC  ; Store continue Addr. for exp. RD NXM
     00F30800'EF    61    D0  3435  3264  MOVL (R1),UNI_MAP  ; Load Unibus Map Register with PFN
                52    60    D0  343C  3265  MOVL (R0),R2   ; Load Test Address
                      62    D4  343F  3266  CLRL (R2)   ; Write to Test Address
                53    60    D0  3441  3267  MOVL (R0),R3   ; Load Test Address
     53    FFFFFE00  8F    CA  3444  3268  BICL2 # XFFFFFE00,R3  ; Set up Unibus Address for UET
     00FFF460 EF    53    B0  344B  3269  MOVW R3,UET_ADDR  ; Load UET with Unibus Address
                      56    D4  3452  3270 20$: CLRL R6   ; Zero counter
     00FFF464 EF    05    B0  3454  3271  MOVW #5,UET_CNTL  ; Perform UET Write
  F8 56    000000FF  8F    F3  345B  3272 30$: AOBLEQ # XFF,R6,30$  ; Allow time for UET write
                      62    D5  3463  3273  TSTL (R2)   ; Reference Test Address again
      1F D59B CF    00    E1  3465  3274 40$: BBC #0,NXM_RD_FLAG,60$ ; If RD NXM Mach Chk, continue test
  14 0000FE00 EF    1F    E1  346B  3275  BBC #31,APT_CTRL_FLAGS,50$ ; Branch if not APT mode
  0000FE44 EF    87  AF    3E  3473  3276  MOVAW B 50$,APT_ERR_PC ; Save Error PC
     0000FE40 EF    01    88  347B  3277  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                D7FD    30  3482  3278  BSBW APT_COMMAND
                  03    11  3485  3279  BRB 60$
                        00  3487  3280 50$: HALT    ; ERROR, referenced Cache location
```

```
                        3488  3281       ; was not invalidated.
                        3488  3282       ; R2 contains referenced address
                        3488  3283       ; R3 contains UET Unibus address
                        3488  3284       ; R1 contains PFN for Unibus ref.
          C8    11      3488  3285  BRB 20$    ; Loop on error
50        04    C0      348A  3286 60$: ADDL2 #4,R0    ; Point to next Cache Test Address
51        04    C0      348D  3287  ADDL2 #4,R1    ; Point to next Unibus PFN
          54    D6      3490  3288  INCL R4
05        54    91      3492  3289  CMPB R4,#5
          03    14      3495  3290  BGTR 65$
       FF79     31      3497  3291  BRW 10$
12        00    DA      349A  3292 65$: MTPR #0,#IPL    ; Lower IPL to allow Interrupts
        0000349D  349D  3293  ENDTST = .
        FFFFFFAD  349D  3294  .IF GE<ENDTST-BGNTST- XF0> ; Check test length
                  349D  3295  .ERROR ; MAXIMUM TEST LENGTH OF F0 HEX BYTES EXCEEDED!
                  349D  3296  .ENDC     ; End conditional
```

```
                                349D  3298  .SBTTL  TEST 14:  LOCKED READ TEST
                                349D  3299
                                349D  3300 ;++
                                349D  3301 ; FUNCTIONAL DESCRIPTION:
                                349D  3302 ;
                                349D  3303 ; This test checks that the Read portion of a Locked Read
                                349D  3304 ; Instruction forces a Cache miss, and gets the data from
                                349D  3305 ; main memory.
                                349D  3306 ;
                                349D  3307 ; The test code first makes a write reference to EFFXXX
                                349D  3308 ; which will result in a Write NXM Interrupt to occur.
                                349D  3309 ; The referenced location, however, will be cached. A Read
                                349D  3310 ; Interlocked instructions is then performed to that same
                                349D  3311 ; location. The test is successful if the reference does
                                349D  3312 ; not go to Cache but goes to main memory where an expected
                                349D  3313 ; Read NXM Machine check will occur. This is verified by checking
                                349D  3314 ; for a clear NXM_RD_FLAG. The test fails if the reference
                                349D  3315 ; goes to cache where it hits and doesn't have to go to main
                                349D  3316 ; memory. This is verified by checking that the RD_NXM_FLAG
                                349D  3317 ; is still set.
                                349D  3318 ;
                                349D  3319 ;
                                349D  3320 ;--
                                349D  3321
                                349D  3322  SEQ  $TN    ; Macro call for Test Sequence Check
                                349D        TEST_14::     ; Continued on next line
          0000FE50 EF    0E    B0  349D         MOVW #14,APT_TEST_NUM ; Load the Test Number
               0000FE52 EF    B6  34A4         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50 EF    B1  34AA         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52 EF        34B0          APT_TEST_COUNT  ; Check for the correct test number
                        20    13  34B5         BEQL T14_S1    ; Sequence correct, continue test
          0000FE00 EF    1F    E1  34B7         BBC #31,APT_CTRL_FLAGS,-
                        17        34BE          T14_HLT    ; Branch if not APT mode
0000FE44 EF    000034D6 EF    9E  34BF         MOVAB T14_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01    B0  34CA         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                      D7AE    30  34D1         BSBW APT_COMMAND  ; Call APT routine
                        01    11  34D4         BRB T14_S1    ; Continue
                                  34D6        T14_HLT:
                              00  34D6         HALT    ; ERROR!!! Program out of sequence
                                  34D7        T14_S1:
           D52A CF    FD AF    DE  34D7         MOVAL T14_S1,RET_PC  ; Store return PC
                                  34DD
                        05    11  34DD  3323  BRB STRT14    ; Go to start of test
                                  34DF  3324  .ALIGN LONG    ; Index value must be long aligned
                     00000000     34E0  3325 INDEX4:  .LONG 0    ; Reserve test location
               50    F9 AF    DE  34E4  3326 STRT14: MOVAL INDEX4,R0  ; Store location
          50    00FFF000 8F    CA  34E8  3327  BICL2 # XFFF000,R0  ; Retain Index value
                   25    00    DA  34EF  3328  MTPR #0,#CADR  ; Flush the Cache
                   27    0F    DA  34F2  3329  MTPR # XF,#CAER  ; Clear the CAER
              D50A CF    01    88  34F5  3330  BISB2 #1,NXM_WT_FLAG  ; Set flag for expected write to NXM
          50    00EFF000 8F    C8  34FA  3331  BISL2 # XEFF000,R0  ; Load NXM Address
                        60    D4  3501  3332  CLRL (R0)    ; Write to NXM
                                  3503  3333  NOP    ; Wait for Interrupt to occur
              D4FC CF    01    88  3504  3334  BISB2 #1,NXM_RD_FLAG  ; Set flag for expected read to NXM
     D4F9 CF    00003515 EF    DE  3509  3335  MOVAL 15$,CONT_PC  ; Store continue Addr. for exp. RD NXM
                   60    00    58  3512  3336 10$: ADAWI #0,(R0)    ; Perform Read Interlock Instr. to NXM
           1F D4EB CF    00    E1  3515  3337 15$: BBC #0,NXM_RD_FLAG,30$ ; If RD NXM Mach Chk., go to next test
```

```
          14 0000FE00 EF    1F   E1  351B  3338  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
          0000FE44 EF   37'AF   3E  3523  3339  MOVAW B 20$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01   88  352B  3340  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          D74D   30  3532  3341  BSBW APT_COMMAND
                            03   11  3535  3342  BRB 30$
                                 00  3537  3343 20$: HALT    ; ERROR, Read ref. of Interlock instr.
                                     3538  3344       ; did not force Cache miss.
                                     3538  3345       ; R0 contains referenced location.
                            D8   11  3538  3346  BRB 10$   ; Loop on error
                          0000   31  353A  3347 30$: BRW TEST_15   ; Go to next test
```

```
                                353D  3348 ;
                                353D  3349 ; NOTE:
                                353D  3350 ; All tests that make use of the TB DATA IPR to write and read
                                353D  3351 ; PTE's in the TB will use the P0BR to load the VA.
                                353D  3352 ;
                                353D  3353
                                353D  3354
                                353D  3355
                                353D  3356  .SBTTL  TEST 15:  TB CONTROL LOGIC TEST
                                353D  3357 ;++
                                353D  3358 ; FUNCTIONAL DESCRIPTION:
                                353D  3359 ;
                                353D  3360 ; Subtests 1-3 check TB maintenance register bits for read/write
                                353D  3361 ; capability.  Subtests 4-9 will attempt simple hits and misses
                                353D  3362 ; in TB Group 0, and subtests 10-15 will attempt same in TB
                                353D  3363 ; Group 1. Subtests 16 and 17 will check for TB multiple hits.
                                353D  3364 ; All subtests will employ the use of the TB DATA
                                353D  3365 ; IPR to write and read PTE s at specified VA's without having
                                353D  3366 ; to enable Memory Mapping.
                                353D  3367 ;
                                353D  3368 ;
                                353D  3369 ;
                                353D  3370 ;--
                                353D  3371
                                353D  3372
                                353D  3373  SEQ  $TN    ; Macro Call for Test Sequence Check
                                353D        TEST_15::     ; Continued on next line
          0000FE50'EF    0F    B0  353D         MOVW #15,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  3544         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  354A         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52 EF        3550          APT_TEST_COUNT  ; Check for the correct test number
                          20    13  3555         BEQL T15_S1    ; Sequence correct, continue test
          0000FE00'EF    1F    E1  3557         BBC #31,APT_CTRL_FLAGS,-
                          17        355E          T15_HLT    ; Branch if not APT mode
0000FE44 EF    00003576 EF    9E  355F         MOVAB T15_HLT,APT_ERR_PC ; Save the error PC
          0000FE40'EF    01    B0  356A         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       D70E    30  3571         BSBW APT_COMMAND  ; Call APT routine
                          01    11  3574         BRB T15_S1    ; Continue
                                    3576        T15_HLT:
                                00  3576         HALT     ; ERROR!!! Program out of sequence
                                    3577        T15_S1:
           D48A CF    FD AF    DE  3577         MOVAL T15_S1,RET_PC  ; Store return PC
                                    357D
                     25    00    DA  357D  3374  MTPR #0,#CADR  ; Enable Cache
                     26    0F    DA  3580  3375  MTPR # XF,#MCESR  ; Clear Mach. Chk. Error Register
                     50    26    DB  3583  3376  MFPR #MCESR,R0  ; Load MCESR into R0
                     50    00    91  3586  3377  CMPB #0,R0    ; Check if MCESR is cleared
                           1F    13  3589  3378  BEQL T15_S2    ; Go to next test
        14 0000FE00'EF     1F    E1  358B  3379  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
        0000FE44 EF    A7 AF    3E  3593  3380  MOVAW B 10$,APT_ERR_PC ; Save Error PC
           0000FE40 EF    01    88  359B  3381  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       D6DD    30  35A2  3382  BSBW APT_COMMAND
                          03    11  35A5  3383  BRB 50$
                                00  35A7  3384 10$: HALT     ; ERROR, unable to clear MCESR
                                    35A8  3385       ; R0 contains the contents of MCESR
                          CD    11  35A8  3386  BRB T15_S1    ; Loop on error
                                    35AA  3387 50$:
```

```
                D457 CF    FD AF    DE   35AA  3388 T15_S2: MOVAL T15_S2,RET_PC  ; Store return PC
                       24    0E    DA   35B0  3389  MTPR # XE,#TBDR  ; Set all possible bits in the TBDR
                       50    24    DB   35B3  3390  MFPR #TBDR,R0  ; Load TBDR into R0
                       50    0E    91   35B6  3391  CMPB # XE,R0   ; Check if all bits set in TBDR
                             1F    13   35B9  3392  BEQL T15_S2A   ; Go to next test
             14 0000FE00 EF    1F    E1   35BB  3393  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
             0000FE44 EF    D7'AF    3E   35C3  3394  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                0000FE40 EF    01    88   35CB  3395  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                             D6AD    30   35D2  3396  BSBW APT_COMMAND
                               03    11   35D5  3397  BRB 50$
                                     00   35D7  3398 10$: HALT    ; ERROR, can't set bits in the TBDR
                                          35D8  3399      ; R0 contains the contents of the TBDR
                               D0    11   35D8  3400  BRB T15_S2   ; Loop on error
                                          35DA  3401 50$:
                D427 CF    CD AF    DE   35DA  3402 T15_S2A:MOVAL T15_S2,RET_PC  ; Store return PC
                       24    0D    DA   35E0  3403  MTPR # XD,#TBDR  ; Set other bit in TBDR
                       50    24    DB   35E3  3404  MFPR #TBDR,R0  ; Load TBDR into R0
                       50    0D    91   35E6  3405  CMPB # XD,R0   ; Check if all bits set in TBDR
                             1F    13   35E9  3406  BEQL T15_S3   ; Go to next test
             14 0000FE00 EF    1F    E1   35EB  3407  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
             0000FE44 EF    07'AF    3E   35F3  3408  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                0000FE40 EF    01    88   35FB  3409  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                             D67D    30   3602  3410  BSBW APT_COMMAND
                               03    11   3605  3411  BRB 50$
                                     00   3607  3412 10$: HALT    ; ERROR, can't set bits in the TBDR
                                          3608  3413      ; R0 contains the contents of the TBDR
                               D0    11   3608  3414  BRB T15_S2A   ; Loop on error
                                          360A  3415 50$:
                D3F7 CF    FD AF    DE   360A  3416 T15_S3: MOVAL T15_S3,RET_PC  ; Store return PC
                       24    00    DA   3610  3417  MTPR #0,#TBDR  ; Clear all bits in the TBDR
                       50    24    DB   3613  3418  MFPR #TBDR,R0  ; Load TBDR into R0
                       50    00    91   3616  3419  CMPB #0,R0   ; Check if all bits clear in TBDR
                             1F    13   3619  3420  BEQL T15_S4   ; Go to next test
             14 0000FE00 EF    1F    E1   361B  3421  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
             0000FE44 EF    37 AF    3E   3623  3422  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                0000FE40 EF    01    88   362B  3423  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                             D64D    30   3632  3424  BSBW APT_COMMAND
                               03    11   3635  3425  BRB 50$
                                     00   3637  3426 10$: HALT    ; ERROR, can't clear bits in the TBDR
                                          3638  3427      ; R0 contains the contents of the TBDR
                               D0    11   3638  3428  BRB T15_S3   ; Loop on error
                                          363A  3429 50$:
                                          363A  3430 ;
                                          363A  3431 ; Subtest 4 checks that a TB hit in Group 0 can be verified by writing and
                                          363A  3432 ; reading a PTE with only the Valid Bit set.
                                          363A  3433 ;
                                          363A  3434
                D3C7 CF    FD AF    DE   363A  3435 T15_S4: MOVAL T15_S4,RET_PC  ; Store return PC
                       26    0F    DA   3640  3436  MTPR # XF,#MCESR  ; Clear Mach. Chk. Error Register
                       39    00    DA   3643  3437  MTPR #0,#TBIA  ; Flush the TB
                       24    0A    DA   3646  3438 5$: MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                       08    00    DA   3649  3439  MTPR #0,#P0BR  ; Load VA=0
                       3A    00    DA   364C  3440  MTPR #0,#TBIS  ; Invalidate TB VA 0
                3B    80000000 8F    DA   364F  3441  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                       51    3B    DB   3656  3442  MFPR #TB_DATA,R1  ; Read PTE into R1
                     1F 51    1F    E0   3659  3443  BBS #TB_VAL,R1,T15_S5 ; Check for TB hit (Valid Bit set)
             14 0000FE00 EF    1F    E1   365D  3444  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
```

```
                0000FE44 EF    79 AF    3E  3665  3445  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF    01    88  366D  3446  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                 D60B    30  3674  3447  BSBW APT_COMMAND
                                   03    11  3677  3448  BRB 50$
                                         00  3679  3449 10$: HALT    ; ERROR, did not get TB Hit
                                             367A  3450       ; R1 bit<31> contains Valid bit
                                   CA    11  367A  3451  BRB 5$    ; Loop on error
                                             367C  3452 50$:
                                             367C  3453 ;
                                             367C  3454 ; Subtest 5 checks that a TB miss in Group 0 can be verified by writing and
                                             367C  3455 ; reading a PTE with the Valid Bit clear.
                                             367C  3456 ;
                                             367C  3457
                     D385 CF    FD AF    DE  367C  3458 T15_S5: MOVAL T15_S5,RET_PC  ; Store return PC
                             26    0F    DA  3682  3459  MTPR #^XF,#MCESR  ; Clear MCESR
                             39    00    DA  3685  3460  MTPR #0,#TBIA  ; Flush the TB
                             24    0A    DA  3688  3461 5$: MTPR #^XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                             08    00    DA  368B  3462  MTPR #0,#P0BR  ; Load VA=0
                             3A    00    DA  368E  3463  MTPR #0,#TBIS  ; Invalidate TB VA 0
                             3B    00    DA  3691  3464  MTPR #0,#TB_DATA  ; Write PTE=0 into TB
                             51    3B    DB  3694  3465  MFPR #TB_DATA,R1  ; Read PTE into R1
                          1F 51    1F    E1  3697  3466  BBC #TB_VAL,R1,T15_S6 ; Check for TB miss (Valid Bit clear)
                14 0000FE00'EF    1F    E1  369B  3467  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
                0000FE44 EF    B7'AF    3E  36A3  3468  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                   0000FE40'EF    01    88  36AB  3469  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                 D5CD    30  36B2  3470  BSBW APT_COMMAND
                                   03    11  36B5  3471  BRB 50$
                                         00  36B7  3472 10$: HALT    ; ERROR, did not get TB miss
                                             36B8  3473       ; R1 bit<31> contains Valid bit
                                   CE    11  36B8  3474  BRB 5$    ; Loop on error
                                             36BA  3475 50$:
                                             36BA  3476 ;
                                             36BA  3477 ; Subtest 6 checks that all bits of the TB Tag Store in Group 0 can be set
                                             36BA  3478 ; by checking for a TB hit with a VA=7FFF0000.
                                             36BA  3479 ;
                                             36BA  3480
                      D347 CF    FD AF    DE  36BA  3481 T15_S6: MOVAL T15_S6,RET_PC  ; Store return PC
                             26    0F    DA  36C0  3482  MTPR #^XF,#MCESR  ; Clear MCESR
                             39    00    DA  36C3  3483  MTPR #0,#TBIA  ; Flush the TB
                             24    0A    DA  36C6  3484 5$: MTPR #^XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                          08    7FFF0000 8F    DA  36C9  3485  MTPR #^X7FFF0000,#P0BR ; Load VA Tag Store = all ones
                          3A    7FFF0000 8F    DA  36D0  3486  MTPR #^X7FFF0000,#TBIS ; Invalidate TB VA 7FFF0000
                          3B    80000000 8F    DA  36D7  3487  MTPR #^X80000000,#TB_DATA ; Write PTE=80000000 into TB
                                   51    3B    DB  36DE  3488  MFPR #TB_DATA,R1  ; Read PTE into R1
                                1F 51    1F    E0  36E1  3489  BBS #TB_VAL,R1,T15_S7 ; Check for TB hit (Valid bit set)
                14 0000FE00 EF    1F    E1  36E5  3490  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
                0000FE44 EF    01 AF    3E  36ED  3491  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF    01    88  36F5  3492  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                 D583    30  36FC  3493  BSBW APT_COMMAND
                                   03    11  36FF  3494  BRB 50$
                                         00  3701  3495 10$: HALT    ; ERROR, did not get TB hit
                                             3702  3496       ; R1 bit<31> contains Valid bit
                                   C2    11  3702  3497  BRB 5$    ; Loop on error
                                             3704  3498 50$:
                                             3704  3499 ;
                                             3704  3500 ; Subtest 7 checks that all bits of the VA Tag Store in Group 0 can be
                                             3704  3501 ; cleared by checking for a TB hit at VA=0.
```

```
                                3704  3502 ;
                                3704  3503
          D2FD CF    FD AF    DE  3704  3504 T15_S7: MOVAL T15_S7,RET_PC  ; Store return PC
                  26    0F    DA  370A  3505  MTPR # XF,#MCESR  ; Clear MCESR
                  39    00    DA  370D  3506  MTPR #0,#TBIA  ; Flush the TB
                  24    0A    DA  3710  3507 5$: MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                  08    00    DA  3713  3508  MTPR #0,#P0BR  ; Load VA=0
                  3A    00    DA  3716  3509  MTPR #0,#TBIS  ; Invalidate TB VA 0
             3B   80000000  8F    DA  3719  3510  MTPR # ^X80000000,#TB_DATA ; Write PTE=80000000 into TB
                  51    3B    DB  3720  3511  MFPR #TB_DATA,R1  ; Read PTE into R1
                1F 51    1F    E0  3723  3512  BBS #TB_VAL,R1,T15_S8 ; Check for TB hit (Valid bit set)
          14 0000FE00'EF    1F    E1  3727  3513  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
          0000FE44 EF    43'AF    3E  372F  3514  MOVAW B 10$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88  3737  3515  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       D541    30  373E  3516  BSBW APT_COMMAND
                          03    11  3741  3517  BRB 50$
                                00  3743  3518 10$: HALT    ; ERROR, did not get TB hit
                                3744  3519      ; R1 bit<31> contains Valid bit
                          CA    11  3744  3520  BRB 5$   ; Loop on error
                                3746  3521 50$:
                                3746  3522 ;
                                3746  3523 ; Subtest 8 checks that all bits of the TB Data Store in Groun 0 can
                                3746  3524 ; be set by writing and reading a PTE=FC007FFF.
                                3746  3525 ;
                                3746  3526
          D2BB CF    FD AF    DE  3746  3527 T15_S8: MOVAL T15_S8,RET_PC  ; Store return PC
                  26    0F    DA  374C  3528  MTPR # XF,#MCESR  ; Clear MCESR
                  39    00    DA  374F  3529  MTPR #0,#TBIA  ; Flush the TB
                  24    0A    DA  3752  3530 5$: MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                  08    00    DA  3755  3531  MTPR #0,#P0BR  ; Load VA=0
                  3A    00    DA  3758  3532  MTPR #0,#TBIS  ; Invalidate TB VA 0
             51   FC007FFF  8F    DA  375B  3533  MOVL # XFC007FFF,R1  ; Load PTE=all ones into R1
                  3B    51    DA  3762  3534  MTPR R1,#TB_DATA  ; Write PTE into TB
                  52    3B    DB  3765  3535  MFPR #TB_DATA,R2  ; Read PTE into R2
             52   03FF8000  8F    CA  3768  3536  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15> (indeterminate)
                53   52    51    CD  376F  3537  XORL3 R1,R2,R3  ; R2 Contains expected PTE-FC007FFF
                           1F    13  3773  3538  BEQL T15_S9  ; Go to next test
          14 0C00FE00 EF    1F    E1  3775  3539  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
          0000FE44 EF    91 AF    3E  377D  3540  MOVAW B 10$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88  3785  3541  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       D4F3    30  378C  3542  BSBW APT_COMMAND
                          03    11  378F  3543  BRB 50$
                                00  3791  3544 10$: HALT    ; ERROR, PTE incorrect
                                3792  3545      ; R1 contains PTE written=FC007FFF
                                3792  3546      ; R2 contains PTE read
                                3792  3547      ; R3 contains bad PTE bit positions
                          BE    11  3792  3548  BRB 5$   ; Loop on error
                                3794  3549 50$:
                                3794  3550 ;
                                3794  3551 ; Subtest 9 checks that all bits of the TB Data Store in Group 0 can
                                3794  3552 ; be cleared by writing and reading a PTE=80000000.
                                3794  3553 ;
                                3794  3554
          D26D CF    FD AF    DE  3794  3555 T15_S9: MOVAL T15_S9,RET_PC  ; Store return PC
                  26    0F    DA  379A  3556  MTPR # XF,#MCESR  ; Clear MCESR
                  39    00    DA  379D  3557  MTPR #0,#TBIA  ; Flush the TB
                  24    0A    DA  37A0  3558 5$: MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
```

```
                08    00    DA   37A3   3559  MTPR #0,#POBR  ; Load VA=0
                3A    00    DA   37A6   3560  MTPR #0,#TBIS  ; Invalidate TB VA 0
          51   80000000 8F    D0   37A9   3561  MOVL # X80000000,R1  ; Load PTE=80000000 into R1
                3B    51    DA   37B0   3562  MTPR R1,#TB_DATA  ; Write PTE into TB
                52    3B    DB   37B3   3563  MFPR #TB_DATA,R2  ; Read PTE into R2
          52   03FF8000 8F    CA   37B6   3564  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15> (indeterminate)
             53   52    51    CD   37BD   3565  XORL3 R1,R2,R3  ; R2 contains expected PTE=80000000
                      1F    13   37C1   3566  BEQL T15_S10  ; Go to next test
       14 0000FE00'EF    1F    E1   37C3   3567  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
       0000FE44 EF    DF AF    3E   37CB   3568  MOVAW B 10$,APT_ERR_PC ; Save Error PC
          0000FE40'EF    01    88   37D3   3569  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      D4A5    30   37DA   3570  BSBW APT_COMMAND
                      03    11   37DD   3571  BRB 50$
                      00    37DF   3572 10$: HALT  ; ERROR, PTE incorect
                            37E0   3573      ; R1 contains PTE written=80000000
                            37E0   3574      ; R2 contains PTE read
                            37E0   3575      ; R3 contains bad PTE bit positions
                      BE    11   37E0   3576  BRB 5$  ; Loop on error
                            37E2   3577 50$:
                            37E2   3578 ;
                            37E2   3579 ; Subtest 10 checks that a TB hit in Group 1 can be verified by writing
                            37E2   3580 ; and reading a PTE with only the Valid Bit set.
                            37E2   3581 ;
                            37E2   3582
          D21F CF    FD AF    DE   37E2   3583 T15_S10:MOVAL T15_S10,RET_PC  ; Store return PC
                26    0F    DA   37E8   3584  MTPR # XF,#MCESR  ; Clear MCESR
                39    00    DA   37EB   3585  MTPR #0,#TBIA  ; Flush the TB
                24    0D    DA   37EE   3586 5$: MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
                08    00    DA   37F1   3587  MTPR #0,#POBR  ; Load VA=0
                3A    00    DA   37F4   3588  MTPR #0,#TBIS  ; Invalidate TB VA 0
          3B   80000000 8F    DA   37F7   3589  MTPR # X80000000,#TB_DATA  ; Write PTE=80000000 into TB
                51    3B    DB   37FE   3590  MFPR #TB_DATA,R1  ; Read PTE into R1
             1F 51    1F    E0   3801   3591  BBS #TB_VAL,R1,T15_S11 ; Check for TB hit (Valid Bit set)
       14 0000FE00 EF    1F    E1   3805   3592  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
       0000FE44 EF    21'AF    3E   380D   3593  MOVAW B 10$,APT_ERR_PC ; Save Error PC
          0000FE40 EF    01    88   3815   3594  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      D463    30   381C   3595  BSBW APT_COMMAND
                      03    11   381F   3596  BRB 50$
                      00    3821   3597 10$: HALT  ; ERROR, did not get TB Hit
                            3822   3598      ; R1 bit<31> contains Valid bit
                      CA    11   3822   3599  BRB 5$  ; Loop on error
                            3824   3600 50$:
                            3824   3601 ;
                            3824   3602 ; Subtest 11 checks that a TB miss in Group 1 can be verified by writing
                            3824   3603 ; and reading a PTE with the Valid Bit clear.
                            3824   3604 ;
                            3824   3605
          D1DD CF    FD AF    DE   3824   3606 T15_S11:MOVAL T15_S11,RET_PC  ; Store return PC
                26    0F    DA   382A   3607  MTPR # XF,#MCESR  ; Clear MCESR
                39    00    DA   382D   3608  MTPR #0,#TBIA  ; Flush the TB
                24    0D    DA   3830   3609 5$: MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
                08    00    DA   3833   3610  MTPR #0,#POBR  ; Load VA=0
                3A    00    DA   3836   3611  MTPR #0,#TBIS  ; Invalidate TB VA 0
                3B    00    DA   3839   3612  MTPR #0,#TB_DATA  ; Write PTE=0 into TB
                51    3B    DB   383C   3613  MFPR #TB_DATA,R1  ; Read PTE into R1
             1F 51    1F    E1   383F   3614  BBC #TB_VAL,R1,T15_S12 ; Check for TB miss (Valid Bit clear)
       14 0000FE00 EF    1F    E1   3843   3615  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
```

```
                0000FE44 EF    5F'AF    3E   384B   3616  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF    01    88   3853   3617  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                D425    30   385A   3618  BSBW APT_COMMAND
                                  03    11   385D   3619  BRB 50$
                                        00   385F   3620 10$: HALT    ; ERROR, did not get TB miss
                                             3860   3621      ; R1 bit<31> contains Valid bit
                                  CE    11   3860   3622  BRB 5$   ; Loop on error
                                             3862   3623 50$:
                                             3862   3624 ;
                                             3862   3625 ; Subtest 12 checks that all bits of the TB Tag Store Group 1 can be set
                                             3862   3626 ; by checking for a TB hit with a VA=7FFF0000.
                                             3862   3627 ;
                                             3862   3628
                    D19F CF    FD AF    DE   3862   3629 T15_S12:MOVAL T15_S12,RET_PC  ; Store return PC
                            26    0F    DA   3868   3630  MTPR #XF,#MCESR  ; Clear MCESR
                            39    00    DA   386B   3631  MTPR #0,#TBIA  ; Flush the TB
                            24    0D    DA   386E   3632 5$: MTPR #XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
                   08    7FFF0000 8F    DA   3871   3633  MTPR #X7FFF0000,#P0BR ; Load VA Tag Store = all ones
                   3A    7FFF0000 8F    DA   3878   3634  MTPR #X7FFF0000,#TBIS ; Invalidate TB VA 7FFF0000
                   3B    80000000 8F    DA   387F   3635  MTPR #X80000000,#TB_DATA ; Write PTE=80000000 into TB
                            51    3B    DB   3886   3636  MFPR #TB_DATA,R1  ; Read PTE into R1
                         1F 51    1F    E0   3889   3637  BBS #TB_VAL,R1,T15_S13 ; Check for TB hit (Valid bit set)
                14 0000FE00 EF    1F    E1   388D   3638  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
                0000FE44 EF    A9'AF    3E   3895   3639  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF    01    88   389D   3640  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                D3DB    30   38A4   3641  BSBW APT_COMMAND
                                  03    11   38A7   3642  BRB 50$
                                        00   38A9   3643 10$: HALT    ; ERROR, did not get TB hit
                                             38AA   3644      ; R1 bit<31> contains Valid bit
                                  C2    11   38AA   3645  BRB 5$   ; Loop on error
                                             38AC   3646 50$:
                                             38AC   3647 ;
                                             38AC   3648 ; Subtest 13 checks that all bits of the TB Tag Store Group 1 can be
                                             38AC   3649 ; cleared by checking for a TB hit with a  VA=0.
                                             38AC   3650 ;
                                             38AC   3651
                    D155 CF    FD AF    DE   38AC   3652 T15_S13:MOVAL T15_S13,RET_PC  ; Store return PC
                            26    0F    DA   38B2   3653  MTPR #XF,#MCESR  ; Clear MCESR
                            39    00    DA   38B5   3654  MTPR #0,#TBIA  ; Flush the TB
                            24    0D    DA   38B8   3655 5$: MTPR #XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
                            08    00    DA   38BB   3656  MTPR #0,#P0BR  ; Load VA=0
                            3A    00    DA   38BE   3657  MTPR #0,#TBIS  ; Invalidate TB VA 0
                   3B    80000000 8F    DA   38C1   3658  MTPR #X80000000,#TB_DATA ; Write PTE=80000000 into TB
                            51    3B    DB   38C8   3659  MFPR #TB_DATA,R1  ; Read PTE into R1
                         1F 51    1F    E0   38CB   3660  BBS #TB_VAL,R1,T15_S14 ; Check for TB hit (Valid bit set)
                14 0000FE00 EF    1F    E1   38CF   3661  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
                0000FE44 EF    EB AF    3E   38D7   3662  MOVAW B 10$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF    01    88   38DF   3663  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                D399    30   38E6   3664  BSBW APT_COMMAND
                                  03    11   38E9   3665  BRB 50$
                                        00   38EB   3666 10$: HALT    ; ERROR, did not get TB hit
                                             38EC   3667      ; R1 bit<31> contains Valid bit
                                  CA    11   38EC   3668  BRB 5$   ; Loop on error
                                             38EE   3669 50$:
                                             38EE   3670 ;
                                             38EE   3671 ; Subtest 14 checks that all bits of the TB Data Store Group 1 can be
                                             38EE   3672 ; set by writing and reading a PTE=FC007FFF.
```

```
                        38EE  3673 ;
                        38EE  3674
     D113 CF    FD AF    DE  38EE  3675 T15_S14:MOVAL T15_S14,RET_PC  ; Store return PC
             26    0F    DA  38F4  3676  MTPR # XF,#MCESR  ; Clear MCESR
             39    00    DA  38F7  3677  MTPR #0,#TBIA  ; Flush the TB
             24    0D    DA  38FA  3678 5$: MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
             08    00    DA  38FD  3679  MTPR #0,#P0BR  ; Load VA=0
             3A    00    DA  3900  3680  MTPR #0,#TBIS  ; Invalidate TB VA 0
     51   FC007FFF 8F    D0  3903  3681  MOVL # XFC007FFF,R1  ; Load PTE=all ones into TB
             3B    51    DA  390A  3682  MTPR R1,#TB_DATA  ; Write PTE into TB
             52    3B    DB  390D  3683  MFPR #TB_DATA,R2  ; Read PTE into R2
     52   03FF8000 8F    CA  3910  3684  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15> (indeterminate)
          53   52    51    CD  3917  3685  XORL3 R1,R2,R3  ; R2 contains expected PTE=FC007FFF
                     1F    13  391B  3686  BEQL T15_S15  ; Go to next test
  14 0000FE00'EF     1F    E1  391D  3687  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
 0000FE44 EF    39'AF     3E  3925  3688  MOVAW B 10$,APT_ERR_PC ; Save Error PC
    0000FE40'EF     01    88  392D  3689  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                  D34B    30  3934  3690  BSBW APT_COMMAND
                    03    11  3937  3691  BRB 50$
                          00  3939  3692 10$: HALT  ; ERROR, PTE incorrect
                              393A  3693      ; R1 contains PTE written =FC007FFF
                              393A  3694      ; R2 contains PTE read
                              393A  3695      ; R3 contains bad PTE bit positions
                    BE    11  393A  3696  BRB 5$  ; Loop on error
                              393C  3697 50$:
                              393C  3698 ;
                              393C  3699 ; Subtest 15 checks that all bits of the TB Data Store Group 1 can be
                              393C  3700 ; cleared by writing and reading a PTE=80000000.
                              393C  3701 ;
                              393C  3702
     D0C5 CF    FD AF    DE  393C  3703 T15_S15:MOVAL T15_S15,RET_PC  ; Store return PC
             26    0F    DA  3942  3704  MTPR # XF,#MCESR  ; Clear MCESR
             39    00    DA  3945  3705  MTPR #0,#TBIA  ; Flush the TB
             24    0D    DA  3948  3706 5$: MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
             08    00    DA  394B  3707  MTPR #0,#P0BR  ; Load VA=0
             3A    00    DA  394E  3708  MTPR #0,#TBIS  ; Invalidate TB VA 0
     51   80000000 8F    D0  3951  3709  MOVL # X80000000,R1  ; Load PTE=80000000 into R1
             3B    51    DA  3958  3710  MTPR R1,#TB_DATA  ; Write PTE into TB
             52    3B    DB  395B  3711  MFPR #TB_DATA,R2  ; Read PTE into R2
     52   03FF8000 8F    CA  395E  3712  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15> (indeterminate)
          53   52    51    CD  3965  3713  XORL3 R1,R2,R3  ; R2 contains expected PTE=80000000
                     1F    13  3969  3714  BEQL 20$  ; Go to next test
  14 0000FE00 EF     1F    E1  396B  3715  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
 0000FE44 EF    87 AF     3E  3973  3716  MOVAW B 10$,APT_ERR_PC ; Save Error PC
    0000FE40 EF     01    88  397B  3717  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                  D2FD    30  3982  3718  BSBW APT_COMMAND
                    03    11  3985  3719  BRB 20$
                          00  3987  3720 10$: HALT  ; ERROR, PTE incorrect
                              3988  3721      ; R1 contains PTE written=80000000
                              3988  3722      ; R2 contains PTE read
                              3988  3723      ; R3 contains bad PTE bit positions
                    BE    11  3988  3724  BRB 5$  ; Loop on error
                              398A  3725 20$:  ; Go to next test
                              398A  3726 ;
                              398A  3727 ; Subtest 16 checks for the occurence of TB multiple hits
                              398A  3728 ; while forcing replacement to Group 0, and not
                              398A  3729 ; disabling either group. A PTE is then written
```

```
                                398A  3730 ; at VA=0 and checked for a hit. A miss indicates that a
                                398A  3731 ; hit was not detected or multiple hits occured.
                                398A  3732 ;
                                398A  3733
     D077 CF    FD AF    DE     398A  3734 T15_S16:MOVAL T15_S16,RET_PC  ; Store return PC
             26    0F    DA     3990  3735  MTPR #XF,#MCESR  ; Clear MCESR
             39    00    DA     3993  3736  MTPR #0,#TBIA  ; Flush the TB
             24    08    DA     3996  3737 10$: MTPR #8,#TBDR  ; Force replace Group 0
             3A    00    DA     3999  3738  MTPR #0,#TBIS  ; Invalidate TB VA 0
             08    00    DA     399C  3739  MTPR #0,#P0BR  ; Load VA = 0
    3B   80000000 8F    DA     399F  3740  MTPR #X80000000,#TB_DATA ; Write PTE=80000000 into TB
             51    3B    DB     39A6  3741  MFPR #TB_DATA,R1  ; Read PTE into R1
          1F 51    1F    E0     39A9  3742  BBS #TB_VAL,R1,T15_S17 ; Check for TB hit (Valid Bit set)
 14 0000FE00'EF    1F    E1     39AD  3743  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
 0000FE44 EF    C9'AF    3E     39B5  3744  MOVAW B 20$,APT_ERR_PC ; Save Error PC
    0000FE40'EF    01    88     39BD  3745  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                 D2BB    30     39C4  3746  BSBW APT_COMMAND  ;
                   03    11     39C7  3747  BRB 30$   ;
                         00     39C9  3748 20$: HALT    ; ERROR, did not get TB hit
                                39CA  3749      ; or received multiple hits
                                39CA  3750      ; R1, bit<31> contains Valid bit
                   CA    11     39CA  3751  BRB 10$   ; Loop on error
                                39CC  3752 30$:
                                39CC  3753 ;
                                39CC  3754 ; Subtest 17 checks for the occurence of TB multiple hits
                                39CC  3755 ; while forcing replacement to Group 1, and not
                                39CC  3756 ; disabling either group. A PTE is then written
                                39CC  3757 ; at VA=0 and checked for a hit. A miss indicates that a
                                39CC  3758 ; hit was not detected or multiple hits occured.
                                39CC  3759 ;
                                39CC  3760
     D035 CF    FD AF    DE     39CC  3761 T15_S17:MOVAL T15_S17,RET_PC  ; Store return PC
             26    0F    DA     39D2  3762  MTPR #XF,#MCESR  ; Clear MCESR
             39    00    DA     39D5  3763  MTPR #0,#TBIA  ; Flush the TB
             24    0C    DA     39D8  3764 10$: MTPR #XC,#TBDR  ; Force replace Group 0
             3A    00    DA     39DB  3765  MTPR #0,#TBIS  ; Invalidate TB VA 0
             08    00    DA     39DE  3766  MTPR #0,#P0BR  ; Load VA = 0
    3B   80000000 8F    DA     39E1  3767  MTPR #X80000000,#TB_DATA ; Write PTE=80000000 into TB
             51    3B    DB     39E8  3768  MFPR #TB_DATA,R1  ; Read PTE into R1
          1F 51    1F    E0     39EB  3769  BBS #TB_VAL,R1,30$  ; Check for TB hit (Valid Bit set)
 14 0000FE00'EF    1F    E1     39EF  3770  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
 0000FE44 EF    0B AF    3E     39F7  3771  MOVAW B 20$,APT_ERR_PC ; Save Error PC
    0000FE40 EF    01    88     39FF  3772  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                 D279    30     3A06  3773  BSBW APT_COMMAND  ;
                   03    11     3A09  3774  BRB 30$   ;
                         00     3A0B  3775 20$: HALT    ; ERROR, did not get TB hit
                                3A0C  3776      ; or received multiple hits
                                3A0C  3777      ; R1, bit<31> contains Valid bit
                   CA    11     3A0C  3778  BRB 10$   ; Loop on error
                                3A0E  3779 30$:
```

```
                                3A0E  3780  .SBTTL  TEST 16:  ADDRESS PATH TAG MEMORY TEST GROUP 0
                                3A0E  3781 ;++
                                3A0E  3782 ; FUNCTIONAL DESCRIPTION:
                                3A0E  3783 ;
                                3A0E  3784 ; This test checks the input and output paths of Group 0's TB
                                3A0E  3785 ; Tag Store for bits S-A-0, S-A-1, and shorted together. This
                                3A0E  3786 ; is accomplished by forcing replacement of TB Group 0 and
                                3A0E  3787 ; referencing TB Index location zero VA<31,15:09>=0, while
                                3A0E  3788 ; writing a minimal bit fault pattern set to the TB Tag Store
                                3A0E  3789 ; (VA<30:16>) and checking for hits. The parity generator,
                                3A0E  3790 ; checker, and hit/miss comparitors of Group 0 are implicitely
                                3A0E  3791 ; checked in this test.
                                3A0E  3792 ;
                                3A0E  3793 ;
                                3A0E  3794 ; IMPLICIT INPUTS:
                                3A0E  3795 ;
                                3A0E  3796 ; TB_PAT_A:: .LONG  X55550000 TB pattern 1
                                3A0E  3797 ;    .LONG  X2AAA0000 TB pattern 2
                                3A0E  3798 ;    .LONG  X33330000 TB pattern 3
                                3A0E  3799 ;    .LONG   X0F0F0000 TB pattern 4
                                3A0E  3800 ;    .LONG  X00FF0000 TB pattern 5
                                3A0E  3801 ;
                                3A0E  3802 ;
                                3A0E  3803 ;
                                3A0E  3804 ;--
                                3A0E  3805
                                3A0E  3806
                                3A0E  3807  SEQ  $TN    ; Macro Call for Test Sequence Check
                                3A0E        TEST_16::     ; Continued on next line
        0000FE50'EF    10   B0  3A0E         MOVW #16,APT_TEST_NUM ; Load the Test Number
             0000FE52'EF   B6   3A15         INCW APT_TEST_COUNT  ; Bump up the test count
             0000FE50'EF   B1   3A1B         CMPW APT_TEST_NUM,-  ; Continued on next line
             0000FE52'EF        3A21          APT_TEST_COUNT  ; Check for the correct test number
                           20   3A26         BEQL T16_S1    ; Sequence correct, continue test
        0000FE00 EF    1F   E1  3A28         BBC #31,APT_CTRL_FLAGS,-
                           17   3A2F          T16_HLT    ; Branch if not APT mode
0000FE44'EF    00003A47 EF   9E  3A30         MOVAB T16_HLT,APT_ERR_PC ; Save the error PC
        0000FE40 EF    01   B0  3A3B         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       D23D 30  3A42         BSBW APT_COMMAND  ; Call APT routine
                         01 11  3A45         BRB T16_S1    ; Continue
                            3A47        T16_HLT:
                         00 3A47         HALT    ; ERROR!!! Program out of sequence
                            3A48        T16_S1:
         CFB9 CF     FD AF   DE  3A48         MOVAL T16_S1,RET_PC  ; Store return PC
                            3A4E
                 39    00   DA  3A4E  3808  MTPR #0,#TBIA  ; Flush the TB
                 24    0A   DA  3A51  3809  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
            54   C898 CF   DE  3A54  3810  MOVAL TB_PAT_A,R4  ; Get address of TB test pattern
                 53    01   D0  3A59  3811  MOVL #1,R3   ; Initialize loop counter
                 08    84   DA  3A5C  3812 10$: MTPR (R4)+,#P0BR  ; Load VA=TB pattern(I)
                 55    08   DB  3A5F  3813  MFPR #P0BR,R5  ; Get copy of VA
                 26    0F   DA  3A62  3814 20$: MTPR # XF,#MCESR  ; Clear MCESR
                 3A    55   DA  3A65  3815  MTPR R5,#TBIS  ; Invalidate TB VA under test
        3B   80000000 8F   DA  3A68  3816  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                 51    3B   DB  3A6F  3817  MFPR #TB_DATA,R1  ; Read PTE into R1
               1F 51   1F   E0  3A72  3818  BBS #TB_VAL,R1,40$  ; Check for TB hit (Valid bit set)
      14 0000FE00 EF   1F   E1  3A76  3819  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
```

```
          0000FE44'EF    92'AF    3E   3A7E  3820  MOVAW B 30$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88   3A86  3821  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         D1F2    30   3A8D  3822  BSBW APT_COMMAND
                           03    11   3A90  3823  BRB 40$
                                 00   3A92  3824 30$: HALT    ; ERROR, didn t get TB hit at spec. VA
                                      3A93  3825       ; R1 bit<31> contains Valid bit
                                      3A93  3826       ; R5 contains VA
                           CD    11   3A93  3827  BRB 20$   ; Loop on error
                      C3 53      05   F3   3A95  3828 40$: AOBLEQ #5,R3,10$  ; Repeat for five patterns
                                      3A99  3829       ; and go to next test.
```

```
                                3A99  3830  .SBTTL  TEST 17:  ADDRESS PATH TAG MEMORY TEST GROUP 1
                                3A99  3831 ;++
                                3A99  3832 ; FUNCTIONAL DESCRIPTION:
                                3A99  3833 ;
                                3A99  3834 ; This test checks the input and outpur paths of Group 1 s TB
                                3A99  3835 ; Tag Store for bits S-A-0, S-A-1, and shorted together. This
                                3A99  3836 ; accomplished by forcing replacement of TB Group 1 and
                                3A99  3837 ; referencing TB Index location zero VA<31,15:09>=0, while
                                3A99  3838 ; writing a minimal bit fault pattern set to the TB Tag Store
                                3A99  3839 ; (VA<30:16>) and checking for hits.  The parity generator,
                                3A99  3840 ; checker, and hit/miss comparitors of Group 1 are implicitely
                                3A99  3841 ; checked in this test.
                                3A99  3842 ;
                                3A99  3843 ;
                                3A99  3844 ; IMPLICIT INPUTS:
                                3A99  3845 ;
                                3A99  3846 ; TB_PAT_A:: .LONG  X55550000 TB pattern 1
                                3A99  3847 ;    .LONG  X2AAA0000 TB pattern 2
                                3A99  3848 ;    .LONG  X33330000 TB pattern 3
                                3A99  3849 ;    .LONG  X0F0F0000 TB pattern 4
                                3A99  3850 ;    .LONG  X00FF0000 TB pattern 5
                                3A99  3851 ;
                                3A99  3852 ;
                                3A99  3853 ;
                                3A99  3854 ;--
                                3A99  3855
                                3A99  3856
                                3A99  3857  SEQ  $TN    ; Macro Call for Test Sequence Check
                                3A99        TEST_17::     ; Continued on next line
        0000FE50'EF    11    B0  3A99         MOVW #17,APT_TEST_NUM ; Load the Test Number
             0000FE52'EF    B6  3AA0         INCW APT_TEST_COUNT  ; Bump up the test count
             0000FE50 EF    B1  3AA6         CMPW APT_TEST_NUM,-  ; Continued on next line
             0000FE52'EF        3AAC          APT_TEST_COUNT  ; Check for the correct test number
                        20  13  3AB1         BEQL T17_S1    ; Sequence correct, continue test
        0000FE00 EF    1F  E1  3AB3         BBC #31,APT_CTRL_FLAGS,-
                        17     3ABA          T17_HLT    ; Branch if not APT mode
0000FE44 EF    00003AD2'EF    9E  3ABB         MOVAB T17_HLT,APT_ERR_PC ; Save the error PC
        0000FE40 EF    01  B0  3AC6         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                      D1B2  30  3ACD         BSBW APT_COMMAND  ; Call APT routine
                        01  11  3AD0         BRB T17_S1    ; Continue
                            3AD2        T17_HLT:
                        00  3AD2         HALT    ; ERROR!!! Program out of sequence
                            3AD3        T17_S1:
         CF2E CF    FD AF    DE  3AD3         MOVAL T17_S1,RET_PC  ; Store return PC
                            3AD9
                39  00  DA  3AD9  3858  MTPR #0,#TBIA  ; Flush the TB
                24  0D  DA  3ADC  3859  MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
            54  C80D CF  DE  3ADF  3860  MOVAL TB_PAT_A,R4  ; Get address of TB test pattern
                53  01  D0  3AE4  3861  MOVL #1,R3   ; Initialize loop counter
                08  84  DA  3AE7  3862 10$: MTPR (R4)+,#P0BR  ; Load TB pattern(I)
                55  08  DB  3AEA  3863  MFPR #P0BR,R5  ; Get copy of VA
                26  0F  DA  3AED  3864 20$: MTPR # XF,#MCESR  ; Clear MCESR
                3A  55  DA  3AF0  3865  MTPR R5,#TBIS  ; Invalidate VA under test
        3B   80000000  8F  DA  3AF3  3866  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                51  3B  DB  3AFA  3867  MFPR #TB_DATA,R1  ; Read PTE into R1
              1F 51  1F  E0  3AFD  3868  BBS #TB_VAL,R1,40$  ; Check for TB hit (Valid bit set)
      14 0000FE00 EF   1F  E1  3B01  3869  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
```

```
          0000FE44'EF   1D'AF    3E  3B09  3870  MOVAW B 30$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88  3B11  3871  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          D167    30  3B18  3872  BSBW APT_COMMAND
                            03    11  3B1B  3873  BRB 40$
                                  00  3B1D  3874 30$: HALT    ; ERROR, didn t get TB hit at spec. VA
                                      3B1E  3875       ; R1 bit<31> contains Valid bit
                                      3B1E  3876       ; R5 contains VA
                            CD    11  3B1E  3877  BRB 20$    ; Loop on error
                      C3 53     05    F3  3B20  3878 40$: AOBLEQ #5,R3,10$  ; Repeat for five patterns
                                      3B24  3879
                          0000    31  3B24  3880  BRW TEST_18   ; and go to next test
```

```
                                3B27  3881  .SBTTL  TEST 18:  TB DATA PATH DATA MEMORY TEST GROUP 0
                                3B27  3882 ;++
                                3B27  3883 ; FUNCTIONAL DESCRIPTION:
                                3B27  3884 ;
                                3B27  3885 ; This test checks the input and output paths of Group 0's
                                3B27  3886 ; TB Data Store for bits S-A-0, S-A-1, shorted together.
                                3B27  3887 ; This is accomplished by forcing replacement of TB Group 0
                                3B27  3888 ; and referencing TB Index location zero (VA=0), and writing
                                3B27  3889 ; a minimal bit fault pattern set into the PTE, and checking
                                3B27  3890 ; for the correct data results.  This test also checks the TB
                                3B27  3891 ; Data Store Output Mux implicitely.
                                3B27  3892 ;
                                3B27  3893 ;
                                3B27  3894 ; IMPLICIT INPUTS:
                                3B27  3895 ;
                                3B27  3896 ; TB_PAT_B:: .LONG  XD4005555 TB pattern 1
                                3B27  3897 ;    .LONG  XA8002AAA TB pattern 2
                                3B27  3898 ;    .LONG  XCC003333 TB pattern 3
                                3B27  3899 ;    .LONG  XBC000F0F TB pattern 4
                                3B27  3900 ;    .LONG  XFC0000FF TB pattern 5
                                3B27  3901 ;
                                3B27  3902 ;
                                3B27  3903 ;--
                                3B27  3904
                                3B27  3905
                                3B27  3906  SEQ  $TN    ; Macro Call for Test Sequence Check
                                3B27        TEST_18::     ; Continued on next line
          0000FE50 EF    12    B0  3B27         MOVW #18,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  3B2E         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  3B34         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        3B3A          APT_TEST_COUNT  ; Check for the correct test number
                           20    13  3B3F         BEQL T18_S1    ; Sequence correct, continue test
          0000FE00 EF    1F    E1  3B41         BBC #31,APT_CTRL_FLAGS,-
                           17        3B48          T18_HLT    ; Branch if not APT mode
     0000FE44 EF    00003B60'EF    9E  3B49         MOVAB T18_HLT,APT_ERR_PC ; Save the error PC
          0000FE40'EF    01    B0  3B54         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       D124    30  3B5B         BSBW APT_COMMAND  ; Call APT routine
                           01    11  3B5E         BRB T18_S1    ; Continue
                                3B60        T18_HLT:
                               00  3B60         HALT    ; ERROR!!! Program out of sequence
                                3B61        T18_S1:
           CEA0 CF    FD AF    DE  3B61         MOVAL T18_S1,RET_PC  ; Store return PC
                                3B67
                     39    00  DA  3B67  3907  MTPR #0,#TBIA   ; Flush the TB
                     24    0A  DA  3B6A  3908  MTPR # XA,#TBDR   ; Replace Grp. 0, disable Grp. 1
                54   C793 CF    DE  3B6D  3909  MOVAL TB_PAT_B,R4   ; Get address of TB test pattern
                     55    01  D0  3B72  3910  MOVL #1,R5    ; Initialize loop counter
                     51    84  D0  3B75  3911 10$: MOVL (R4)+,R1   ; Load PTE=TB Pattern(I) into R1
                     26    0F  DA  3B78  3912 20$: MTPR # XF,#MCESR   ; Clear MCESR
                     3A    00  DA  3B7B  3913  MTPR #0,#TBIS   ; Invalidate TB VA 0
                     08    00  DA  3B7E  3914  MTPR #0,#P0BR   ; Load VA=0
                     3B    51  DA  3B81  3915  MTPR R1,#TB_DATA   ; Write PTE into TB
                     52    3B  DB  3B84  3916  MFPR #TB_DATA,R2   ; Read PTE into R2
          52    03FF8000 8F    CA  3B87  3917  BICL2 #PTE_MASK,R2   ; Clear PTE<25:15>, (indeterminate)
               53   52    51  CD  3B8E  3918  XORL3 R1,R2,R3   ; R2 contains expected PTE
                           1F    13  3B92  3919  BEQL 40$    ; Continue test
          14 0000FE00'EF    1F    E1  3B94  3920  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
```

```
          0000FE44'EF   B0'AF    3E  3B9C  3921  MOVAW B 30$,APT_ERR_PC ; Save Error PC
             0000FE40 EF   01    88  3BA4  3922  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        DOD4    30  3BAB  3923  BSBW APT_COMMAND
                          03    11  3BAE  3924  BRB 40$
                                00  3BB0  3925 30$: HALT    ; ERROR, PTE incorrect
                                    3BB1  3926       ; R1 contains PTE written
                                    3BB1  3927       ; R2 contains PTE read
                                    3BB1  3928       ; R3 contains bad PTE bit positions
                          C5    11  3BB1  3929  BRB 20$   ; Loop on error
                    BE 55   05    F3  3BB3  3930 40$: AOBLEQ #5,R5,10$  ; Repeat for five patterns
                                    3BB7  3931       ; and go to next test
```

```
                     3BB7  3932  .SBTTL  TEST 19:  TB DATA PATH DATA MEMORY TEST GROUP 1
                     3BB7  3933 ;++
                     3BB7  3934 ; FUNCTIONAL DESCRIPTION:
                     3BB7  3935 ;
                     3BB7  3936 ; This test checks the input and output paths of Group 1 s
                     3BB7  3937 ; TB Data Store for bits S-A-0, S-A-1, and shorted together.
                     3BB7  3938 ; This is accomplished by forcing replacement of TB Group 1
                     3BB7  3939 ; and referencing TB Index location 0 (VA=0), and writing a
                     3BB7  3940 ; minimal bit fault pattern set into the PTE, and checking for
                     3BB7  3941 ; the correct data results.  This test also checks the TB Data
                     3BB7  3942 ; Store Output Mux implicitely.
                     3BB7  3943 ;
                     3BB7  3944 ;
                     3BB7  3945 ; IMPLICIT INPUTS:
                     3BB7  3946 ;
                     3BB7  3947 ; TB_PAT_B:: .LONG  XD4005555 TB pattern 1
                     3BB7  3948 ;    .LONG  XA8002AAA TB pattern 2
                     3BB7  3949 ;    .LONG  XCC003333 TB pattern 3
                     3BB7  3950 ;    .LONG  XBC000F0F TB pattern 4
                     3BB7  3951 ;    .LONG  XFC0000FF TB pattern 5
                     3BB7  3952 ;
                     3BB7  3953 ;
                     3BB7  3954 ;
                     3BB7  3955 ;
                     3BB7  3956 ;--
                     3BB7  3957
                     3BB7  3958
                     3BB7  3959  SEQ  $TN    ; Macro Call for Test Sequence Check
                     3BB7        TEST_19::     ; Continued on next line
          0000FE50'EF   13   B0   3BB7          MOVW #19,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF   B6   3BBE          INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF   B1   3BC4          CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        3BCA           APT_TEST_COUNT  ; Check for the correct test number
                          20   13   3BCF          BEQL T19_S1    ; Sequence correct, continue test
          0000FE00 EF     1F   E1   3BD1          BBC #31,APT_CTRL_FLAGS,-
                          17        3BD8           T19_HLT    ; Branch if not APT mode
0000FE44 EF    00003BF0 EF   9E   3BD9          MOVAB T19_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF     01   B0   3BE4          MOVW #1,APT_ERR_FLAG  ; Set the error flag
                        D094   30   3BEB          BSBW APT_COMMAND  ; Call APT routine
                          01   11   3BEE          BRB T19_S1    ; Continue
                               3BF0        T19_HLT:
                          00   3BF0          HALT     ; ERROR!!! Program out of sequence
                               3BF1        T19_S1:
           CE10 CF      FD AF   DE   3BF1          MOVAL T19_S1,RET_PC  ; Store return PC
                               3BF7
                    39    00   DA   3BF7  3960    MTPR #0,#TBIA  ; Flush the TB
                    24    0D   DA   3BFA  3961    MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
               54   C703  CF   DE   3BFD  3962    MOVAL TB_PAT_B,R4  ; Get address of TB test pattern
                    55    01   D0   3C02  3963    MOVL #1,R5    ; Initialize loop counter
                    51    84   D0   3C05  3964 10$: MOVL (R4)+,R1  ; Load PTE=TB Pattern(I) into R1
                    26    0F   DA   3C08  3965 20$: MTPR # XF,#MCESR  ; Clear MCESR
                    3A    00   DA   3C0B  3966    MTPR #0,#TBIS  ; Invalidate TB VA 0
                    08    00   DA   3C0E  3967    MTPR #0,#P0BR  ; Load VA=0
                    3B    51   DA   3C11  3968    MTPR R1,#TB_DATA  ; Write PTE into TB
                    52    3B   DB   3C14  3969    MFPR #TB_DATA,R2  ; Read PTE into R2
          52   03FF800C  8F   CA   3C17  3970    BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
               53   52    51   CD   3C1E  3971    XORL3 R1,R2,R3  ; R2 contains expected PTE
```

```
                      1F   13  3C22  3972  BEQL 40$    ; Continue test
          14 0000FE00'EF   1F  E1  3C24  3973  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
          0000FE44'EF  40'AF  3E  3C2C  3974  MOVAW B 30$,APT_ERR_PC ; Save Error PC
             0000FE40 EF   01  88  3C34  3975  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        D044  30  3C3B  3976  BSBW APT_COMMAND
                          03  11  3C3E  3977  BRB 40$
                              00  3C40  3978 30$: HALT     ; ERROR, PTE incorrect
                                  3C41  3979       ; R1 contains PTE written
                                  3C41  3980       ; R2 contains PTE read
                                  3C41  3981       ; R3 contains bad PTE bit positions
                          C5  11  3C41  3982  BRB 20$    ; Loop on error
                    BE 55   05  F3  3C43  3983 40$: AOBLEQ #5,R5,10$  ; Repeat for five patterns and
                                  3C47  3984       ; go to next test
```

```
                                3C47  3985  .SBTTL  TEST 20:  TB HIT/MISS COMPARITORS TEST GROUP 0
                                3C47  3986 ;++
                                3C47  3987 ; FUNCTIONAL DESCRIPTION:
                                3C47  3988 ;
                                3C47  3989 ; This test will verify the inputs to the Hit/Miss Comparitors
                                3C47  3990 ; (= s checkers) of TB Group 0 by referencing TB Index location
                                3C47  3991 ; zero (VA=0), and alternately floating a one across a field of
                                3C47  3992 ; zeroes on each side of the comparitors while checking for
                                3C47  3993 ; misses. The Hit/Miss Comparitors were implicitely checked for
                                3C47  3994 ; hits by virtue of TB tests 16 and 17.
                                3C47  3995 ;
                                3C47  3996 ;
                                3C47  3997 ;
                                3C47  3998 ;--
                                3C47  3999
                                3C47  4000
                                3C47  4001  SEQ  $TN    ; Macro Call for Test Sequence Check
                                3C47        TEST_20::     ; Continued on next line
          0000FE50'EF    14    B0  3C47         MOVW #20,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  3C4E         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  3C54         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        3C5A          APT_TEST_COUNT  ; Check for the correct test number
                         20    13  3C5F         BEQL T20_S1    ; Sequence correct, continue test
          0000FE00 EF    1F    E1  3C61         BBC #31,APT_CTRL_FLAGS,-
                         17        3C68          T20_HLT    ; Branch if not APT mode
     0000FE44 EF   00003C80'EF    9E  3C69         MOVAB T20_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01    B0  3C74         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       D004    30  3C7B         BSBW APT_COMMAND  ; Call APT routine
                         01    11  3C7E         BRB T20_S1    ; Continue
                             3C80        T20_HLT:
                             00  3C80         HALT     ; ERROR!!! Program out of sequence
                             3C81        T20_S1:
      CD80 CF    FD AF    DE  3C81         MOVAL T20_S1,RET_PC  ; Store return PC
                             3C87
                39    00    DA  3C87  4002  MTPR #0,#TBIA  ; Flush the TB
                24    0A    DA  3C8A  4003  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
     54   00010000 8F    D0  3C8D  4004  MOVL # X10000,R4  ; Load Test Address1 into R4
                55    00    D0  3C94  4005  MOVL #0,R5    ; Load Test Address2 into R5
                5(    01    D0  3C97  4006  MOVL #1,R6    ; Initialize loop counter
                26    0F    DA  3C9A  4007 10$: MTPR # XF,#MCESR  ; Clear MCESR
                3A    54    DA  3C9D  4008  MTPR R4,#TBIS  ; Invalidate TB VA
                08    54    DA  3CA0  4009  MTPR R4,#P0BR  ; Load VA=Test Address1
     3B   80000000 8F    DA  3CA3  4010  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                51    3B    DB  3CAA  4011  MFPR #TB_DATA,R1  ; Read PTE into R1
              1F 51    1F    E0  3CAD  4012  BBS #TB_VAL,R1,30$  ; If TB hit, continue test
  14 0000FE00 EF    1F    E1  3CB1  4013  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
 0000FE44 EF    CD AF    3E  3CB9  4014  MOVAW B 20$,APT_ERR_PC ; Save Error PC
     0000FE40 EF    01    88  3CC1  4015  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                  CFB7    30  3CC8  4016  BSBW APT_COMMAND
                         03    11  3CCB  4017  BRB 30$
                             00  3CCD  4018 20$: HALT     ; ERROR, didn't get TB hit at
                             3CCE  4019       ; Test Address1
                             3CCE  4020       ; R1 bit<31> contains valid bit
                             3CCE  4021       ; R4 contains VA
                         CA    11  3CCE  4022  BRB 10$    ; Loop on error
                08    55    DA  3CD0  4023 30$: MTPR R5,#P0BR  ; Load VA=Test Address2
                51    3B    DB  3CD3  4024  MFPR #TB_DATA,R1  ; Read PTE at VA=Test Address2 into R1
```

```
                1F 51    1F    E1  3CD6  4025  BBC #TB_VAL,R1,50$  ; If TB miss, continue test
          14 0000FE00 EF    1F    E1  3CDA  4026  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
          00C0FE44 EF    F6 AF    3E  3CE2  4027  MOVAW B 40$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88  3CEA  4028  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        CF8E    30  3CF1  4029  BSBW APT_COMMAND
                          03    11  3CF4  4030  BRB 50$
                          00    3CF6  4031 40$: HALT    ; ERROR, did not get TB miss at
                                3CF7  4032      ; VA=Test Address2
                                3CF7  4033      ; R1 bit<31> contains Valid bit
                                3CF7  4034      ; R5 contains VA
                          D7    11  3CF7  4035  BRB 30$   ; Loop on error
                54    54    01    78  3CF9  4036 50$: ASHL #1,R4,R4  ; Shift Test Address1 left by one
                            56    D6  3CFD  4037  INCL R6
                      0F    56    91  3CFF  4038  CMPB R6,#15
                            03    14  3D02  4039  BGTR 55$
                        FF93    31  3D04  4040  BRW 10$
                                3D07  4041 55$:
                                3D07  4042
                 CCFA CF    FD AF    DE  3D07  4043 T20_S2: MOVAL T20_S2,RET_PC  ; Store return PC
                          39    00    DA  3D0D  4044  MTPR #0,#TBIA  ; Flush the TB
                          54    00    D0  3D10  4045  MOVL #0,R4   ; Load Test Address1 into R4
             55   00010000 8F    D0  3D13  4046  MOVL # X10000,R5  ; Load Test Address2 into R5
                          56    01    D0  3D1A  4047  MOVL #1,R6   ; Initialize loop counter
                          26    0F    DA  3D1D  4048 10$: MTPR # XF,#MCESR  ; Clear MCESR
                          3A    54    DA  3D20  4049  MTPR R4,#TBIS  ; Invalidate TB VA
                          08    54    DA  3D23  4050  MTPR R4,#P0BR  ; Load VA=Test Address1
             3B   80000000 8F    DA  3D26  4051  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                          51    3B    DB  3D2D  4052  MFPR #TB_DATA,R1  ; Read PTE into R1
                1F 51    1F    E0  3D30  4053  BBS #TB_VAL,R1,30$  ; If TB hit, continue test
          14 0000FE00 EF    1F    E1  3D34  4054  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
          0000FE44 EF    50 AF    3E  3D3C  4055  MOVAW B 20$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88  3D44  4056  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        CF34    30  3D4B  4057  BSBW APT_COMMAND
                          03    11  3D4E  4058  BRB 30$
                          00    3D50  4059 20$: HALT    ; ERROR, did not get TB hit at
                                3D51  4060      ; VA=Test Address1
                                3D51  4061      ; R1 bit<31> contains Valid Bit
                                3D51  4062      ; R4 contains VA
                          CA    11  3D51  4063  BRB 10$   ; Loop on error
                      08    55    DA  3D53  4064 30$: MTPR R5,#P0BR  ; Load VA=Test Address2
                      51    3B    DB  3D56  4065  MFPR #TB_DATA,R1  ; Read PTE at VA=Test Address2 into R1
                1F 51    1F    E1  3D59  4066  BBC #TB_VAL,R1,50$  ; If TB miss, continue test
          14 0000FE00 EF    1F    E1  3D5D  4067  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
          0000FE44 EF    79 AF    3E  3D65  4068  MOVAW B 40$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88  3D6D  4069  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        CF0B    30  3D74  4070  BSBW APT_COMMAND
                          03    11  3D77  4071  BRB 50$
                          00    3D79  4072 40$: HALT    ; ERROR, did not get TB miss at
                                3D7A  4073      ; VA=Test Address2
                                3D7A  4074      ; R1 bit<31> contains Valid bit
                                3D7A  4075      ; R5 contains VA
                          D7    11  3D7A  4076  BRB 30$   ; Loop on error
                55    55    01    78  3D7C  4077 50$: ASHL #1,R5,R5  ; Shift Test Address2 left by one
                            56    D6  3D80  4078  INCL R6
                      0F    56    91  3D82  4079  CMPB R6,#15
                            03    14  3D85  4080  BGTR 55$
                        FF93    31  3D87  4081  BRW 10$
```

```
                3D8A  4082 55$:
```

```
                                3D8A  4083  .SBTTL  TEST 21:  TB HIT/MISS COMPARITORS TEST GROUP 1
                                3D8A  4084 ;++
                                3D8A  4085 ; FUNCTIONAL DESCRIPTION:
                                3D8A  4086 ;
                                3D8A  4087 ; This test will verify the inputs to the Hit/Miss Comparitors
                                3D8A  4088 ; (='s checkers) of TB Group 1 by referencing TB Index location
                                3D8A  4089 ; zero (VA=0), and alternately floating a one across a field of
                                3D8A  4090 ; zeroes on each side of the comparitors while checking for misses.
                                3D8A  4091 ; The Hit/Miss Comparitors were implicitely checked for hits by
                                3D8A  4092 ; virtue of TB Tests 16 and 17.
                                3D8A  4093 ;
                                3D8A  4094 ;
                                3D8A  4095 ;
                                3D8A  4096 ;--
                                3D8A  4097
                                3D8A  4098
                                3D8A  4099  SEQ  $TN    ; Macro Call for Test Sequence Check
                                3D8A        TEST_21::     ; Continued on next line
          0000FE50 EF    15    B0  3D8A          MOVW #21,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  3D91          INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50 EF    B1  3D97          CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        3D9D           APT_TEST_COUNT  ; Check for the correct test number
                          20   13  3DA2          BEQL T21_S1    ; Sequence correct, continue test
          0000FE00 EF    1F   E1  3DA4          BBC #31,APT_CTRL_FLAGS,-
                          17      3DAB           T21_HLT    ; Branch if not APT mode
     0000FE44'EF    00003DC3 EF   9E  3DAC          MOVAB T21_HLT,APT_ERR_PC ; Save the error PC
               0000FE40'EF    01   B0  3DB7          MOVW #1,APT_ERR_FLAG  ; Set the error flag
                         CEC1   30  3DBE          BSBW APT_COMMAND  ; Call APT routine
                          01    11  3DC1          BRB T21_S1    ; Continue
                                3DC3        T21_HLT:
                                00  3DC3          HALT     ; ERROR!!! Program out of sequence
                                3DC4        T21_S1:
           CC3D CF     FD AF    DE  3DC4          MOVAL T21_S1,RET_PC  ; Store return PC
                                3DCA
                     39    00   DA  3DCA  4100  MTPR #0,#TBIA  ; Flush the TB
                     24    0D   DA  3DCD  4101  MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
               54    00010000 8F   D0  3DD0  4102  MOVL # X10000,R4  ; Load Test Address1 into R4
                     55    00   D0  3DD7  4103  MOVL #0,R5    ; Load Test Address2 into R5
                     56    01   D0  3DDA  4104  MOVL #1,R6    ; Initialize loop counter
                     26    0F   DA  3DDD  4105 10$: MTPR # XF,#MCESR  ; Clear MCESR
                     3A    54   DA  3DE0  4106  MTPR R4,#TBIS  ; Invalidate TB VA
                     08    54   DA  3DE3  4107  MTPR R4,#P0BR  ; Load VA=Test Address1
               3B    80000000 8F   DA  3DE6  4108  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                     51    3B   DB  3DED  4109  MFPR #TB_DATA,R1  ; Read PTE into R1
                   1F 51    1F   E0  3DF0  4110  BBS #TB_VAL,R1,30$  ; If TB hit, continue test
          14 0000FE00 EF    1F   E1  3DF4  4111  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
          0000FE44'EF      10 AF   3E  3DFC  4112  MOVAW B 20$,APT_ERR_PC ; Save Error PC
               0000FE40 EF    01   88  3E04  4113  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         CE74   30  3E0B  4114  BSBW APT_COMMAND
                          03    11  3E0E  4115  BRB 30$
                                00  3E10  4116 20$: HALT     ; ERROR, did not get TB hit at
                                3E11  4117       ; Test Address1
                                3E11  4118       ; R1 bit<31> contains Valid bit
                                3E11  4119       ; R4 contains VA
                          CA    11  3E11  4120  BRB 10$    ; Loop on error
                     08    55   DA  3E13  4121 30$: MTPR R5,#P0BR  ; Load VA=Test Address2
                     51    3B   DB  3E16  4122  MFPR #TB_DATA,R1  ; Read PTE at VA=Test Address2 into R1
```

```
                1F 51    1F    E1   3E19   4123  BBC #TB_VAL,R1,50$  ; If TB miss, continue test
          14 0000FE00'EF    1F    E1   3E1D   4124  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
          0000FE44'EF   39'AF    3E   3E25   4125  MOVAW B 40$,APT_ERR_PC ; Save Error PC
             0000FE40'EF    01    88   3E2D   4126  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       CE4B    30   3E34   4127  BSBW APT_COMMAND
                         03    11   3E37   4128  BRB 50$
                               00   3E39   4129 40$: HALT    ; ERROR, did not get TB miss at
                                    3E3A   4130       ; VA=Test Address2
                                    3E3A   4131       ; R1 bit<31> contains Valid bit
                                    3E3A   4132       ; R5 contains VA
                         D7    11   3E3A   4133  BRB 30$   ; Loop on error
                54   54    01    78   3E3C   4134 50$: ASHL #1,R4,R4  ; Shift Test Address1 left by one
                         56    D6   3E40   4135  INCL R6
                      0F   56    91   3E42   4136  CMPB R6,#15
                         03    14   3E45   4137  BGTR 55$
                       FF93    31   3E47   4138  BRW 10$
                                    3E4A   4139 55$:
                                    3E4A   4140
             CBB6 CF   FEB9 CF    DE   3E4A   4141 T21_S2: MOVAL T20_S2,RET_PC  ; Store return PC
                      39   00    DA   3E51   4142  MTPR #0,#TBIA  ; Flush the TB
                      54   00    D0   3E54   4143  MOVL #0,R4   ; Load Test Address1 into R4
              55   00010000 8F    D0   3E57   4144  MOVL # X10000,R5  ; Load Test Address2 into R5
                      56   01    D0   3E5E   4145  MOVL #1,R6   ; Initialize loop counter
                      26   0F    DA   3E61   4146 10$: MTPR # XF,#MCESR  ; Clear MCESR
                      3A   54    DA   3E64   4147  MTPR R4,#TBIS  ; Invalidate TB VA
                      08   54    DA   3E67   4148  MTPR R4,#P0BR  ; Load VA=Test Address1
           3B   80000000 8F    DA   3E6A   4149  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                      51   3B    DB   3E71   4150  MFPR #TB_DATA,R1  ; Read PTE into R1
                1F 51    1F    E0   3E74   4151  BBS #TB_VAL,R1,30$  ; If TB hit, continue test
          14 0000FE00'EF    1F    E1   3E78   4152  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
          0000FE44'EF   94'AF    3E   3E80   4153  MOVAW B 20$,APT_ERR_PC ; Save Error PC
             0000FE40'EF    01    88   3E88   4154  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       CDF0    30   3E8F   4155  BSBW APT_COMMAND
                         03    11   3E92   4156  BRB 30$
                               00   3E94   4157 20$: HALT    ; ERROR, did not get TB hit at
                                    3E95   4158       ; VA=Test Address1
                                    3E95   4159       ; R1 bit<31> contains Valid Bit
                                    3E95   4160       ; R4 contains VA
                         CA    11   3E95   4161  BRB 10$   ; Loop on error
                      08   55    DA   3E97   4162 30$: MTPR R5,#P0BR  ; Load VA=Test Address2
                      51   3B    DB   3E9A   4163  MFPR #TB_DATA,R1  ; Read PTE at VA=Test Address2 into R1
                1F 51    1F    E1   3E9D   4164  BBC #TB_VAL,R1,50$  ; If TB miss, continue test
          14 0000FE00'EF    1F    E1   3EA1   4165  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
          0000FE44 EF   BD'AF    3E   3EA9   4166  MOVAW B 40$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88   3EB1   4167  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       CDC7    30   3EB8   4168  BSBW APT_COMMAND
                         03    11   3EBB   4169  BRB 50$
                               00   3EBD   4170 40$: HALT    ; ERROR, did not get TB miss at
                                    3EBE   4171       ; VA=Test Address2
                                    3EBE   4172       ; R1 bit<31> contains Valid bit
                                    3EBE   4173       ; R5 contains VA
                         D7    11   3EBE   4174  BRB 30$   ; Loop on error
                55   55    01    78   3EC0   4175 50$: ASHL #1,R5,R5  ; Shift Test Address2 left by one
                         56    D6   3EC4   4176  INCL R6
                      0F   56    91   3EC6   4177  CMPB R6,#15
                         03    14   3EC9   4178  BGTR 55$
                       FF93    31   3ECB   4179  BRW 10$
```

```
                     3ECE  4180 55$:
0000   31            3ECE  4181  BRW TEST_22   ; Go to next test
```

```
                                3ED1  4182  .SBTTL  TEST 22:  TB DUAL ADDRESS TAG STORE TEST GROUP 0
                                3ED1  4183 ;++
                                3ED1  4184 ; FUNCTIONAL DESCRIPTION:
                                3ED1  4185 ;
                                3ED1  4186 ; This test checks the TB Index lines (VA<31,15:09>) to the
                                3ED1  4187 ; Tag Store of Group 0 for bits S-A-0, S-A-1, and shorted
                                3ED1  4188 ; together by referencing 9 Index locations generated by
                                3ED1  4189 ; floating a one across a field of zeroes in the Index.
                                3ED1  4190 ; Zeroes are written into the TB Tag Store at Index location
                                3ED1  4191 ; zero. Ones are then written into the next Index location (200).
                                3ED1  4192 ; Location zero is then read and checked for a hit. If successful,
                                3ED1  4193 ; ones are written into the next Index location (400), and
                                3ED1  4194 ; location zero is read again and checked for a hit. This
                                3ED1  4195 ; process continues for writing ones into all 8 Index locations,
                                3ED1  4196 ; and going back and rereading location zero, and checking for
                                3ED1  4197 ; a hit. This sequence is controlled by an inner loop pointer
                                3ED1  4198 ; and counter. When the inner loop is exited, zeroes are then
                                3ED1  4199 ; written into Index location 200, and ones are written to
                                3ED1  4200 ; all the other Index locations. An outer loop pointer and
                                3ED1  4201 ; counter controls the sequence of writing zeroes, and then
                                3ED1  4202 ; control passes to the inner loop for the writing of ones.
                                3ED1  4203 ;
                                3ED1  4204 ;
                                3ED1  4205 ; IMPLICIT INPUTS:
                                3ED1  4206 ;
                                3ED1  4207 ; TB_PAT_C:: .LONG 0  TB Index location 1
                                3ED1  4208 ;    .LONG  X200  TB Index location 2
                                3ED1  4209 ;    .LONG  X400  TB Index location 3
                                3ED1  4210 ;    .LONG  X800  TB Index location 4
                                3ED1  4211 ;    .LONG  X1000  TB Index location 5
                                3ED1  4212 ;    .LONG   X2000  TB Index location 6
                                3ED1  4213 ;    .LONG  X4000  TB Index location 7
                                3ED1  4214 ;    .LONG  X8000  TB Index location 8
                                3ED1  4215 ;    .LONG  X80000000 TB Index location 9
                                3ED1  4216 ;
                                3ED1  4217 ;
                                3ED1  4218 ;--
                                3ED1  4219
                                3ED1  4220
                                3ED1  4221  SEQ  $TN    ; Macro call for Test Sequence Check
                                3ED1        TEST_22::     ; Continued on next line
          0000FE50'EF    16    B0  3ED1          MOVW #22,APT_TEST_NUM ; Load the Test Number
               0000FE52 EF    B6  3ED8          INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50 EF    B1  3EDE          CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        3EE4           APT_TEST_COUNT  ; Check for the correct test number
                     20    13    3EE9          BEQL T22_S1    ; Sequence correct, continue test
          0000FE00'EF    1F    E1  3EEB          BBC #31,APT_CTRL_FLAGS,-
                     17        3EF2           T22_HLT    ; Branch if not APT mode
0000FE44 EF    00003F0A EF    9E  3EF3          MOVAB T22_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01    B0  3EFE          MOVW #1,APT_ERR_FLAG  ; Set the error flag
                   CD7A    30    3F05          BSBW APT_COMMAND  ; Call APT routine
                     01    11    3F08          BRB T22_S1    ; Continue
                           3F0A        T22_HLT:
                     00    3F0A          HALT     ; ERROR!!! Program out of sequence
                           3F0B        T22_S1:
           CAF6 CF    FD AF    DE  3F0B          MOVAL T22_S1,RET_PC  ; Store return PC
                           3F11
```

```
                26   0F   DA  3F11  4222  MTPR # XF,#MCESR  ; Clear MCESR
                39   00   DA  3F14  4223  MTPR #0,#TBIA  ; Flush the TB
                24   0A   DA  3F17  4224  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp.1
                54   01   D0  3F1A  4225  MOVL #1,R4   ; Initialize outer loop counter
           56   C3F7 CF   DE  3F1D  4226  MOVAL TB_PAT_C,R6  ; Set up test pat pointer, outer loop
                55   01   D0  3F22  4227 10$: MOVL #1,R5   ; Initialize inner loop counter
           57   C3EF CF   DE  3F25  4228  MOVAL TB_PAT_C,R7  ; Set up test pat pointer, inner loop
                67   66   D1  3F2A  4229 20$: CMPL (R6),(R7)  ; Make sure test patterns not equal
                     51   13  3F2D  4230  BEQL 50$   ; If =, get next pat. for inner loop
                08   66   DA  3F2F  4231 30$: MTPR (R6),#P0BR  ; Load VA
      3B   80000000 8F   DA  3F32  4232  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                50   67   D0  3F39  4233  MOVL (R7),R0   ; Store  Test pattern
      50   3FFF0000 8F   C0  3F3C  4234  ADDL2 # X3FFF0000,R0  ; Generate new VA
                08   50   DA  3F43  4235  MTPR R0,#P0BR  ; Load VA
      3B   80000000 8F   DA  3F46  4236  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                08   66   DA  3F4D  4237  MTPR (R6),#P0BR  ; Reload first VA again
                51   3B   DB  3F50  4238  MFPR #TB_DATA,R1  ; Read PTE into R1
             29 51   1F   E0  3F53  4239  BBS #TB_VAL,R1,50$  ; If TB hit, continue test
                53   08   DB  3F57  4240  MFPR #P0BR,R3  ; Store VA
      50   3FFF0000 8F   CA  3F5A  4241  BICL # X3FFF0000,R0  ; Save previous Index value
   14 0000FE00 EF   1F   E1  3F61  4242  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
   0000FE44 EF   7D   AF  3E  3F69  4243  MOVAW B 40$,APT_ERR_PC ; Save Error PC
      0000FE40 EF   01   88  3F71  4244  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                  CD07   30  3F78  4245  BSBW APT_COMMAND
                     03   11  3F7B  4246  BRB 50$
                          00  3F7D  4247 40$: HALT    ; ERROR, did not get TB hit
                              3F7E  4248      ; at specified VA.
                              3F7E  4249      ; R3 contains VA
                              3F7E  4250      ; R1 bit<31> contains Valid Bit
                              3F7E  4251      ; R0 contains previous Index used
                AF   11   3F7E  4252  BRB 30$   ; Loop on error
      57   04   C0   3F80  4253 50$: ADDL2 #4,R7   ; Point to next pattern, inner loop
   A3 55   09   F3   3F83  4254  AOBLEQ #9,R5,20$  ; Continue inner loop til 9 pat. used
      56   04   C0   3F87  4255  ADDL2 #4,R6   ; Point to next pattern, outer loop
                54   D6   3F8A  4256  INCL R4
      09        54   91   3F8C  4257  CMPB R4,#9
                03   14   3F8F  4258  BGTR 55$
                   FF8E   31   3F91  4259  BRW 10$
                          3F94  4260 55$:
```

```
                        3F94  4261  .SBTTL  TEST 23:  TB DUAL ADDRESS TAG STORE TEST GROUP 1
                        3F94  4262 ;**
                        3F94  4263 ; FUNCTIONAL DESCRIPTION:
                        3F94  4264 ;
                        3F94  4265 ; This test checks the TB Index lines (VA<31,15:09>) to the
                        3F94  4266 ; Tag Store of Group 1 for bits S-A-0, S-A-1, and shorted
                        3F94  4267 ; together by referencing 9 Index locations generated by
                        3F94  4268 ; floating a one across a field of zeroes in the Index.
                        3F94  4269 ; Zeroes are written into the TB Tag Store at Index location
                        3F94  4270 ; zero. Ones are then written into the next Index location (200).
                        3F94  4271 ; Location zero is then read and checked for a hit. If successful,
                        3F94  4272 ; ones are written into the next Index location (400), and
                        3F94  4273 ; location zero is read again and checked for a hit. This
                        3F94  4274 ; process continues for writing ones into all 8 Index locations,
                        3F94  4275 ; and going back and rereading location zero, and checking for
                        3F94  4276 ; a hit. This sequence is controlled by an inner loop pointer
                        3F94  4277 ; and counter. When the inner loop is exited, zeroes are then
                        3F94  4278 ; written into Index location 200, and ones are written to
                        3F94  4279 ; all the other Index locations. An outer loop pointer and
                        3F94  4280 ; counter controls the sequence of writing zeroes, and then
                        3F94  4281 ; control passes to the inner loop for the writing of ones.
                        3F94  4282 ;
                        3F94  4283 ;
                        3F94  4284 ; IMPLICIT INPUTS:
                        3F94  4285 ;
                        3F94  4286 ; TB_PAT_C:: .LONG 0  TB Index location 1
                        3F94  4287 ;    .LONG  X200  TB Index location 2
                        3F94  4288 ;    .LONG  X400  TB Index location 3
                        3F94  4289 ;    .LONG  X800  TB Index location 4
                        3F94  4290 ;    .LONG  X1000  TB Index location 5
                        3F94  4291 ;    .LONG   X2000  TB Index location 6
                        3F94  4292 ;    .LONG  X4000  TB Index location 7
                        3F94  4293 ;    .LONG  X8000  TB Index location 8
                        3F94  4294 ;    .LONG  X80000000 TB Index location 9
                        3F94  4295 ;
                        3F94  4296 ;
                        3F94  4297 ;
                        3F94  4298 ;
                        3F94  4299 ;
                        3F94  4300 ;
                        3F94  4301 ;--
                        3F94  4302
                        3F94  4303
                        3F94  4304  SEQ  $TN    ; Macro call for Test Sequence Check
                        3F94       TEST_23::     ; Continued on next line
        0000FE50 EF    17    B0   3F94          MOVW #23,APT_TEST_NUM ; Load the Test Number
             0000FE52 EF    B6   3F9B          INCW APT_TEST_COUNT  ; Bump up the test count
             0000FE50 EF    B1   3FA1          CMPW APT_TEST_NUM,-  ; Continued on next line
             0000FE52 EF         3FA7           APT_TEST_COUNT  ; Check for the correct test number
                      20    13   3FAC          BEQL T23_S1    ; Sequence correct, continue test
        0000FE00 EF    1F    E1   3FAE          BBC #31,APT_CTRL_FLAGS,-
                       17         3FB5           T23_HLT    ; Branch if not APT mode
0000FE44 EF    00003FCD EF    9E   3FB6          MOVAB T23_HLT,APT_ERR_PC ; Save the error PC
        0000FF40 EF    01    B0   3FC1          MOVW #1,APT_ERR_FLAG  ; Set the error flag
                     CCB7    30   3FC8          BSBW APT_COMMAND  ; Call APT routine
                       01    11   3FCB          BRB T23_S1    ; Continue
                             3FCD       T23_HLT:
```

```
                                00  3FCD          HALT    ; ERROR!!! Program out of sequence
                                    3FCE        T23_S1:
          CA33 CF    FD AF     DE  3FCE          MOVAL T23_S1,RET_PC  ; Store return PC
                                    3FD4
                 26    0F     DA  3FD4  4305    MTPR # XF,#MCESR  ; Clear MCESR
                 39    00     DA  3FD7  4306    MTPR #0,#TBIA  ; Flush the TB
                 24    0D     DA  3FDA  4307    MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp.0
                 54    01     D0  3FDD  4308    MOVL #1,R4   ; Initialize outer loop counter
              56   C334 CF     DE  3FE0  4309    MOVAL TB_PAT_C,R6  ; Set up test pat pointer, outer loop
                 55    01     D0  3FE5  4310 10$: MOVL #1,R5   ; Initialize inner loop counter
              57   C32C CF     DE  3FE8  4311    MOVAL TB_PAT_C,R7  ; Set up test pat pointer, inner loop
                 67    66     D1  3FED  4312 20$: CMPL (R6),(R7)  ; Make sure test patterns not equal
                       51     13  3FF0  4313    BEQL 50$   ; If =, get next pat. for inner loop
                 08    66     DA  3FF2  4314 30$: MTPR (R6),#P0BR  ; Load VA
           3B    80000000 8F     DA  3FF5  4315    MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                 50    67     D0  3FFC  4316    MOVL (R7),R0   ; Store Test pattern
           50    3FFF0000 8F     C0  3FFF  4317    ADDL2 # X3FFF0000,R0  ; Generate new VA
                 08    50     DA  4006  4318    MTPR R0,#P0BR  ; Load VA
           3B    80000000 8F     DA  4009  4319    MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                 08    66     DA  4010  4320    MTPR (R6),#P0BR  ; Reload first VA again
                 51    3B     DB  4013  4321    MFPR #TB_DATA,R1  ; Read PTE into R1
              29 51    1F     E0  4016  4322    BBS #TB_VAL,R1,50$  ; If TB hit, continue test
                 53    08     DB  401A  4323    MFPR #P0BR,R3  ; Store VA
           50    3FFF0000 8F     CA  401D  4324    BICL # X3FFF0000,R0  ; Save previous Index value
        14 0000FE00'EF    1F     E1  4024  4325    BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
     0000FE44'EF    40 AF     3E  402C  4326    MOVAW B 40$,APT_ERR_PC ; Save Error PC
        0000FE40'EF    01     88  4034  4327    BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                    CC44     30  403B  4328    BSBW APT_COMMAND
                      03     11  403E  4329    BRB 50$
                             00  4040  4330 40$: HALT    ; ERROR, did not get TB hit
                                 4041  4331       ; at specified VA.
                                 4041  4332       ; R3 contains VA
                                 4041  4333       ; R1 bit<31> contains Valid Bit
                                 4041  4334       ; R0 contains previous Index used
                      AF     11  4041  4335    BRB 30$   ; Loop on error
                 57   04     C0  4043  4336 50$: ADDL2 #4,R7   ; Point to next pattern, inner loop
              A3 55   09     F3  4046  4337    AOBLEQ #9,R5,20$  ; Continue inner loop til 9 pat. used
                 56   04     C0  404A  4338    ADDL2 #4,R6   ; Point to next pattern, outer loop
                      54     D6  404D  4339    INCL R4
                 09   54     91  404F  4340    CMPB R4,#9
                      03     14  4052  4341    BGTR 55$
                    FF8E     31  4054  4342    BRW 10$
                                 4057  4343 55$:
```

```
                                4057  4344  .SBTTL  TEST 24:  TB DUAL ADDRESS DATA STORE TEST GROUP 0
                                4057  4345 ;++
                                4057  4346 ; FUNCTIONAL DESCRIPTION:
                                4057  4347 ;
                                4057  4348 ; This test checks the TB Index lines to the Data Store Rams of
                                4057  4349 ; Group 0 for bits S-A-0, S-A-1, and shorted together by floating
                                4057  4350 ; a one across the Index lines, and writing zeroes into the PTE
                                4057  4351 ; at Index location zero, then writing ones into the PTE at
                                4057  4352 ; Index location 200. Index location zero is then read, and
                                4057  4353 ; the PTE is checked for the correct data. This process is
                                4057  4354 ; continued in the same manner as the TB Dual Address Tag Store
                                4057  4355 ; Tests.
                                4057  4356 ;
                                4057  4357 ;
                                4057  4358 ;
                                4057  4359 ; IMPLICIT INPUTS:
                                4057  4360 ;
                                4057  4361 ; TB_PAT_C::  LONG 0  TB Index location 1
                                4057  4362 ;   .LONG  X200  TB Index location 2
                                4057  4363 ;   .LONG  X400  TB Index location 3
                                4057  4364 ;   .LONG  X800  TB Index location 4
                                4057  4365 ;   .LONG  X1000  TB Index location 5
                                4057  4366 ;   .LONG   X2000  TB Index location 6
                                4057  4367 ;   .LONG  X4000  TB Index location 7
                                4057  4368 ;   .LONG  X8000  TB Index location 8
                                4057  4369 ;   .LONG  X80000000 TB Index location 9
                                4057  4370 ;
                                4057  4371 ;
                                4057  4372 ;
                                4057  4373 ;
                                4057  4374 ;--
                                4057  4375
                                4057  4376
                                4057  4377  SEQ  $TN    ; Macro call for Test Sequence Check
                                4057       TEST_24::     ; Continued on next line
          0000FE50'EF    18    B0  4057         MOVW #24,APT_TEST_NUM ; Load the Test Number
               0000FE52 EF     B6  405E         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF     B1  4064         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52 EF         406A          APT_TEST_COUNT  ; Check for the correct test number
                          20   13  406F         BEQL T24_S1    ; Sequence correct, continue test
          0000FE00'EF    1F    E1  4071         BBC #31,APT_CTRL_FLAGS,-
                          17       4078          T24_HLT    ; Branch if not APT mode
0000FE44 EF    00004090 EF    9E  4079         MOVAB T24_HLT,APT_ERR_PC ; Save the error PC
          0000FE40'EF    01    B0  4084         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       CBF4    30  408B         BSBW APT_COMMAND  ; Call APT routine
                         01    11  408E         BRB T24_S1    ; Continue
                               4090       T24_HLT:
                               00  4090         HALT     ; ERROR!!! Program out of sequence
                               4091       T24_S1:
           C970 CF    FD AF    DE  4091         MOVAL T24_S1,RET_PC  ; Store return PC
                               4097
                    26   0F    DA  4097  4378  MTPR #^XF,#MCESR  ; Clear MCESR
                    39   00    DA  409A  4379  MTPR #0,#TBIA  ; Flush the TB
                    24   0A    DA  409D  4380  MTPR #^XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                    54   01    D0  40A0  4381  MOVL #1,R4    ; Initialize outer loop counter
                56  C271 CF    DE  40A3  4382  MOVAL TB_PAT_C,R6  ; Set up test pat. pointer, outer loop
                    55   01    D0  40A8  4383 10$: MOVL #1,R5    ; Initialize inner loop counter
```

```
                57   C269 CF    DE  40AB  4384  MOVAL TB_PAT_C,R7  ; Set up test pat. pointer, inner loop
                     67   66    D1  40B0  4385 20$: CMPL (R6),(R7)  ; Make sure test patterns not equal
                          4F    13  40B3  4386  BEQL 50$   ; If =, get next pat. for inner loop
                     08   66    DA  40B5  4387 30$: MTPR (R6),#P0BR  ; Load VA
           51    80000000 8F    D0  40B8  4388  MOVL #^X80000000,R1  ; Load PTE into R1
                     3B   51    DA  40BF  4389  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                     08   67    DA  40C2  4390  MTPR (R7),#P0BR  ; Load VA
                     58   08    DB  40C5  4391  MFPR #P0BR,R8  ; Save copy of VA
           3B    FC007FFF 8F    DA  40C8  4392  MTPR #^XFC007FFF,#TB_DATA ; Write PTE=FC007FFF into TB
                     08   66    DA  40CF  4393  MTPR (R6),#P0BR  ; Reload first VA again
                     52   3B    DB  40D2  4394  MFPR #TB_DATA,R2  ; Read PTE into R2
           52    03FF8000 8F    CA  40D5  4395  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
                53   52   51    CD  40DC  4396  XORL3 R1,R2,R3  ; R2 contains expected PTE
                          22    13  40E0  4397  BEQL 50$   ; Continue test
                     59   08    DB  40E2  4398  MFPR #P0BR,R9  ; Save VA
        14 0000FE00 EF    1F    E1  40E5  4399  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
        0000FE44'EF    01'AF    3E  40ED  4400  MOVAW B 40$,APT_ERR_PC ; Save Error PC
           0000FE40 EF    01    88  40F5  4401  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       CB83    30  40FC  4402  BSBW APT_COMMAND
                          03    11  40FF  4403  BRB 50$
                                00  4101  4404 40$: HALT    ; PTE incorrect
                                    4102  4405       ; R9 contains VA
                                    4102  4406       ; R1 contains PTE written
                                    4102  4407       ; R2 contains PTE read
                                    4102  4408       ; R3 contains XOR of PTE's
                                    4102  4409       ; R8 contains previous VA
                                    4102  4410       ; FC007FFF=previous PTE written
                     B1   11    4102  4411  BRB 30$   ; Loop on error
                57   04   C0    4104  4412 50$: ADDL2 #4,R7   ; Point to next pattern, inner loop
        A5      55   09   F3    4107  4413  AOBLEQ #9,R5,20$  ; Continue inner loop til 9 pat. used
                56   04   C0    410B  4414  ADDL2 #4,R6   ; Point to next pattern, outer loop
                     54   D6    410E  4415  INCL R4
                09   54   91    4110  4416  CMPB R4,#9
                     03   14    4113  4417  BGTR 55$
                   FF90   31    4115  4418  BRW 10$
                                4118  4419 55$:
```

```
                                4118  4420  .SBTTL  TEST 25:  TB DUAL ADDRESS DATA STORE TEST GROUP 1
                                4118  4421 ;++
                                4118  4422 ; FUNCTIONAL DESCRIPTION:
                                4118  4423 ;
                                4118  4424 ; This test checks the TB Index lines to the Data Store Rams of
                                4118  4425 ; Group 1 for bits S-A-0, S-A-1, and shorted together by floating
                                4118  4426 ; a one across the Index lines, and writing zeroes into the PTE
                                4118  4427 ; at Index location zero, then writing ones into the PTE at
                                4118  4428 ; Index location 200. Index location zero is then read, and
                                4118  4429 ; the PTE is checked for the correct data. This process is
                                4118  4430 ; continued in the same manner as the TB Dual Address Tag Store
                                4118  4431 ; Tests.
                                4118  4432 ;
                                4118  4433 ;
                                4118  4434 ;
                                4118  4435 ; IMPLICIT INPUTS:
                                4118  4436 ;
                                4118  4437 ; TB_PAT_C:: .LONG 0  TB Index location 1
                                4118  4438 ;    .LONG  X200  TB Index location 2
                                4118  4439 ;    .LONG  X400  TB Index location 3
                                4118  4440 ;    .LONG  X800  TB Index location 4
                                4118  4441 ;    .LONG  X1000  TB Index location 5
                                4118  4442 ;    .LONG   X2000  TB Index location 6
                                4118  4443 ;    .LONG  X4000  TB Index location 7
                                4118  4444 ;    .LONG  X8000  TB Index location 8
                                4118  4445 ;    .LONG  X80000000 TB Index location 9
                                4118  4446 ;
                                4118  4447 ;
                                4118  4448 ;--
                                4118  4449
                                4118  4450
                                4118  4451  SEQ  $TN    ; Macro call for Test Sequence Check
                                4118        TEST_25::     ; Continued on next line
          0000FE50'EF    19    B0  4118         MOVW #25,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  411F         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  4125         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        412B          APT_TEST_COUNT  ; Check for the correct test number
                     20    13  4130         BEQL T25_S1    ; Sequence correct, continue test
          0000FE00'EF    1F    E1  4132         BBC #31,APT_CTRL_FLAGS,-
                     17        4139          T25_HLT    ; Branch if not APT mode
0000FE44 EF    00004151 EF    9E  413A         MOVAB T25_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01    B0  4145         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       CB33    30  414C         BSBW APT_COMMAND  ; Call APT routine
                         01    11  414F         BRB T25_S1    ; Continue
                               4151        T25_HLT:
                               00  4151         HALT    ; ERROR!!! Program out of sequence
                               4152        T25_S1:
           C8AF CF    FD AF    DE  4152         MOVAL T25_S1,RET_PC  ; Store return PC
                               4158
                    26    0F    DA  4158  4452  MTPR #^XF,#MCESR  ; Clear MCESR
                    39    00    DA  415B  4453  MTPR #0,#TBIA  ; Flush the TB
                    24    0D    DA  415E  4454  MTPR #^XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
                    54    01    D0  4161  4455  MOVL #1,R4    ; Initialize outer loop counter
               56   C1B0  CF    DE  4164  4456  MOVAL TB_PAT_C,R6  ; Set up test pat. pointer, outer loop
                    55    01    D0  4169  4457 10$: MOVL #1,R5    ; Initialize inner loop counter
               57   C1A8  CF    DE  416C  4458  MOVAL TB_PAT_C,R7  ; Set up test pat. pointer, inner loop
                    67    66    D1  4171  4459 20$: CMPL (R6),(R7)  ; Make sure test patterns not equal
```

```
                      4F   13  4174  4460  BEQL 50$    ; If =, get next pat. for inner loop
                08    66   DA  4176  4461 30$: MTPR (R6),#P0BR  ; Load VA
          51    80000000 8F   D0  4179  4462  MOVL # X80000000,R1  ; Load PTE into R1
                3B    51   DA  4180  4463  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                08    67   DA  4183  4464  MTPR (R7),#P0BR  ; Load VA
                58    08   DB  4186  4465  MFPR #P0BR,R8  ; Save copy of VA
          3B    FC007FFF 8F   DA  4189  4466  MTPR # XFC007FFF,#TB_DATA ; Write PTE=FC007FFF into TB
                08    66   DA  4190  4467  MTPR (R6),#P0BR  ; Reload first VA again
                52    3B   DB  4193  4468  MFPR #TB_DATA,R2  ; Read PTE into R2
          52    03FF8000 8F   CA  4196  4469  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
             53 52    51   CD  419C  4470  XORL3 R1,R2,R3  ; R2 contains expected PTE
                      22   13  41A1  4471  BEQL 50$    ; Continue test
                59    08   DB  41A3  4472  MFPR #P0BR,R9  ; Save VA
       14 0000FE00 EF    1F   E1  41A6  4473  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
       0000FE44 EF    C2'AF   3E  41AE  4474  MOVAW B 40$,APT_ERR_PC ; Save Error PC
          0000FE40 EF    01   88  41B6  4475  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                     CAC2   30  41BD  4476  BSBW APT_COMMAND
                      03   11  41C0  4477  BRB 50$
                      00   41C2  4478 40$: HALT     ; PTE incorrect
                           41C3  4479       ; R9 contains VA
                           41C3  4480       ; R1 contains PTE written
                           41C3  4481       ; R2 contains PTE read
                           41C3  4482       ; R3 contains XOR of PTE's
                           41C3  4483       ; R8 contains previous VA
                           41C3  4484       ; FC007FFF=previous PTE written
                B1   11  41C3  4485  BRB 30$    ; Loop on error
             57    04   C0  41C5  4486 50$: ADDL2 #4,R7    ; Point to next pattern, inner loop
          A5 55    09   F3  41C8  4487  AOBLEQ #9,R5,20$  ; Continue inner loop til 9 pat. used
             56    04   C0  41CC  4488  ADDL2 #4,R6    ; Point to next pattern, outer loop
                54   D6  41CF  4489  INCL R4
          09    54   91  41D1  4490  CMPB R4,#9
                03   14  41D4  4491  BGTR 55$
                FF90   31  41D6  4492  BRW 10$
                           41D9  4493 55$:
                0000   31  41D9  4494  BRW TEST_26    ; Go to next test.
```

```
                                41DC  4495  .SBTTL  TEST 26:  TB DYNAMIC TEST TAG STORE MEMORY GROUP 0
                                41DC  4496 ;++
                                41DC  4497 ; FUNCTIONAL DESCRIPTION:
                                41DC  4498 ;
                                41DC  4499 ; This test verifies the ability to write and read all of the
                                41DC  4500 ; locations in the Tag Store Rams of TB Group 0 by use of a
                                41DC  4501 ; marching pattern  applied to the VA Tag Store.
                                41DC  4502 ;
                                41DC  4503 ;
                                41DC  4504 ; The test consists of two subtests:
                                41DC  4505 ; Subtest 1 checks all locations whi'e incrementing through the
                                41DC  4506 ; Index.
                                41DC  4507 ; Subtest 2 checks all locations while decrementing through the Index.
                                41DC  4508 ;
                                41DC  4509 ;--
                                41DC  4510
                                41DC  4511
                                41DC  4512  SEQ  $TN    ; Macro call for Test Sequence Check
                                41DC        TEST_26::     ; Continued on next line
          0000FE50'EF   1A   B0 41DC          MOVW #26,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF   B6 41E3          INCW APT_TEST_COUNT   ; Bump up the test count
               0000FE50'EF   B1 41E9          CMPW APT_TEST_NUM,-   ; Continued on next line
               0000FE52'EF      41EF           APT_TEST_COUNT  ; Check for the correct test number
                         20   13 41F4          BEQL T26_S1    ; Sequence correct, continue test
          0000FE00 EF    1F   E1 41F6          BBC #31,APT_CTRL_FLAGS,-
                         17      41FD           T26_HLT    ; Branch if not APT mode
     0000FE44 EF    00004215'EF   9E 41FE          MOVAB T26_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01   B0 4209          MOVW #1,APT_ERR_FLAG  ; Set the error flag
                      CA6F   30 4210          BSBW APT_COMMAND  ; Call APT routine
                        01   11 4213          BRB T26_S1    ; Continue
                             4215         T26_HLT:
                        00   00 4215          HALT    ; ERROR!!! Program out of sequence
                             4216         T26_S1:
           C7EB CF     FD AF   DE 4216          MOVAL T26_S1,RET_PC  ; Store return PC
                             421C
                   26    0F   DA 421C  4513  MTPR # XF,#MCESR  ; Clear MCESR
                   39    00   DA 421F  4514  MTPR #0,#TBIA  ; Flush the TB
                   24    0A   DA 4222  4515  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                             4225  4516
                             4225  4517 ;
                             4225  4518 ; The following code fills all of the TB Tag Store locations with zeroes.
                             4225  4519 ;
                             4225  4520
                   54    01   D0 4225  4521  MOVL #1,R4    ; Initialize counter
                         50   D4 4228  4522  CLRL R0    ; Load initial Index value
                   08    00   DA 422A  4523  MTPR #0,#P0BR  ; Load VA
              3B   80000000 8F   DA 422D  4524 10$: MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                        C7D8   30 4234  4525  BSBW INC_INDEX  ; Go get the next Index value
           EE 54    00000100 8F   F3 4237  4526  AOBLEQ # X100,R4,10$  ; Continue for 256 locations
                             423F  4527
                             423F  4528 ;
                             423F  4529 ; The next section of code reads and checks each Tag Store location
                             423F  4530 ; for VA=0, and then writes all ones into that location.
                             423F  4531 ;
                             423F  4532
                   54    01   D0 423F  4533  MOVL #1,R4    ; Initialize counter
                         50   D4 4242  4534  CLRL R0    ; Load initial Index value
```

```
                    08    00    DA  4244  4535  MTPR #0,#P0BR  ; Load VA
                    51    3B    DB  4247  4536 20$: MFPR #TB_DATA,R1  ; Read PTE into R1
                 22 51    1F    E0  424A  4537  BBS #TB_VAL,R1,40$  ; If TB hit, continue test
                    55    08    DB  424E  4538  MFPR #P0BR,R5  ; Save VA
           14 0000FE00'EF    1F    E1  4251  4539  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
           0000FE44'EF   6D'AF    3E  4259  4540  MOVAW B 30$,APT_ERR_PC ; Save Error PC
              0000FE40 EF    01    88  4261  4541  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       CA17    30  4268  4542  BSBW APT_COMMAND
                          03    11  426B  4543  BRB 40$
                                00  426D  4544 30$: HALT    ; ERROR,
                                    426E  4545       ; did not get TB hit at specified VA
                                    426E  4546       ; R5 contains VA
                                    426E  4547       ; R1 Bit<31> contains Valid Bit
                          D7    11  426E  4548  BRB 20$   ; Loop on error
              50    7FFF0000 8F    C8  4270  4549 40$: BISL2 # X7FFF0000,R0  ; Make VA=all ones
                    08    50    DA  4277  4550  MTPR R0,#P0BR  ; Load VA
              3B    80000000 8F    DA  427A  4551  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
              50    7FFF0000 8F    CA  4281  4552  BICL2 # X7FFF0000,R0  ; Set up to get next Index value
                       C784    30  4288  4553  BSBW INC_INDEX  ; Go get another Index value
           B4 54    00000100 8F    F3  428B  4554  AOBLEQ # X100,R4,20$  ; Continue for 256 locations
                                    4293  4555
                                    4293  4556 ;
                                    4293  4557 ; The next section of code reads and checks each Tag Store location
                                    4293  4558 ; for VA=ones, and then writes all zeroes into that location.
                                    4293  4559 ;
                                    4293  4560
                    54    01    D0  4293  4561  MOVL #1,R4   ; Initialize counter
              50    7FFF0000 8F    D0  4296  4562  MOVL # X7FFF0000,R0  ; Load VA into R0
                    08    50    DA  429D  4563 50$: MTPR R0,#P0BR  ; Load VA
                    51    3B    DB  42A0  4564  MFPR #TB_DATA,R1  ; Read PTE into R1
                 22 51    1F    E0  42A3  4565  BBS #TB_VAL,R1,70$  ; If TB hit, continue test
                    55    08    DB  42A7  4566  MFPR #P0BR,R5  ; Save VA
           14 0000FE00'EF    1F    E1  42AA  4567  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
           0000FE44 EF    C6 AF    3E  42B2  4568  MOVAW B 60$,APT_ERR_PC ; Save Error PC
              0000FE40 EF    01    88  42BA  4569  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       C9BE    30  42C1  4570  BSBW APT_COMMAND
                          03    11  42C4  4571  BRB 70$
                                00  42C6  4572 60$: HALT    ; ERROR,
                                    42C7  4573       ; did not get TB hit at specified VA
                                    42C7  4574       ; R5 contains VA
                                    42C7  4575       ; R1 Bit<31> contains Valid Bit
                          D4    11  42C7  4576  BRB 50$   ; Loop on error
              50    7FFF0000 8F    CA  42C9  4577 70$: BICL2 # X7FFF0000,R0  ; Set up for next Index value
                    08    50    DA  42D0  4578  MTPR R0,#P0BR  ; Load VA
              3B    80000000 8F    DA  42D3  4579  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                       C732    30  42DA  4580  BSBW INC_INDEX  ; Go get next Index value
              50    7FFF0000 8F    C8  42DD  4581  BISL2 # X7FFF0000,R0  ; Set VA=all ones
           B1 54    00000100 8F    F3  42E4  4582  AOBLEQ # X100,R4,50$  ; Continue for 256 locations
                          1A    11  42EC  4583  BRB T26_S2   ; Go to next subtest
                                    42EE  4584
                                    42EE  4585 ;
                                    42EE  4586 ; The next section of code is used only to initialize conditons to
                                    42EE  4587 ; execute T26_S2 in event a scope loop is desired as a result of a
                                    42EE  4588 ; Machine Check Exception. This code shortens the scope loop.
                                    42EE  4589 ;
                                    42EE  4590
                                    42EE  4591 T26_INIT1:    ; Return label for T26_S2 loop on err.
```

```
                    54    01    D0  42EE  4592  MOVL #1,R4    ; Initialize counter
                          50    D4  42F1  4593  CLRL R0    ; Load initial Index value
                    08    00    DA  42F3  4594  MTPR #0,#P0BR   ; Load VA
          3B   80000000  8F    DA  42F6  4595 10$: MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                      C70F    30  42FD  4596  BSBW INC_INDEX  ; Go get the next Index value
          54   00000100  8F    F3  4300  4597  AOBLEQ # X100,R4,10$  ; Continue for 256 locations
                                    4308  4598
                                    4308  4599
                                    4308  4600 ;
                                    4308  4601 ; The next section of code reads and checks each Tag Store location
                                    4308  4602 ; starting from the highest Index location for VA=zeroes and then
                                    4308  4603 ; writes all ones into that location.
                                    4308  4604 ;
                                    4308  4605
           C6F9 CF    E3  AF    DE  4308  4606 T26_S2: MOVAL T26_INIT1,RET_PC ; Store RET PC for T26_S2 loop on err.
                    54    01    D0  430E  4607  MOVL #1,R4    ; Initialize counter
          50   8000FE00  8F    D0  4311  4608  MOVL # X8000FE00,R0   ; Load VA into R0
                    08    50    DA  4318  4609  MTPR R0,#P0BR   ; Load VA
                    51    3B    DB  431B  4610 80$: MFPR #TB_DATA,R1   ; Read PTE into R1
                 22 51    1F    E0  431E  4611  BBS #TB_VAL,R1,100$   ; If TB hit continue test
                    55    08    DB  4322  4612  MFPR #P0BR,R5   ; Save VA
       14 0000FE00'EF    1F    E1  4325  4613  BBC #31,APT_CTRL_FLAGS,90$ ; Branch if not APT mode
       0000FE44'EF   41'AF    3E  432D  4614  MOVAW B 90$,APT_ERR_PC ; Save Error PC
          0000FE40'EF    01    88  4335  4615  BISB2 #1,APT_ERR_FLAG   ; Set Error Flag
                      C943    30  433C  4616  BSBW APT_COMMAND
                         03    11  433F  4617  BRB 100$
                               00  4341  4618 90$: HALT     ; ERROR,
                                   4342  4619       ; did not get TB hit at specified VA
                                   4342  4620       ; R5 contains VA
                                   4342  4621       ; R1 Bit<31> contains Valid Bit
                         D7    11  4342  4622  BRB 80$    ; Loop on error
          50   7FFF0000  8F    C8  4344  4623 100$: BISL2 # X7FFF0000,R0   ; Make VA=all ones
                    08    50    DA  434B  4624  MTPR R0,#P0BR   ; Load VA
          3B   80000000  8F    DA  434E  4625  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
          50   7FFF0000  8F    CA  4355  4626  BICL2 # X7FFF0000,R0   ; Set up for next Index value
                      C6C6    30  435C  4627  BSBW DEC_INDEX  ; Go get another Index value
       B4 54   00000100  8F    F3  435F  4628  AOBLEQ # X100,R4,80$  ; Continue for 256 Index locations
                                    4367  4629
                                    4367  4630 ;
                                    4367  4631 ; The next section for code reads and checks each Tag Store location
                                    4367  4632 ; starting at the highest Index location for a VA=ones and then
                                    4367  4633 ; writes a VA=zero into that location.
                                    4367  4634 ;
                                    4367  4635
                    54    01    D0  4367  4636  MOVL #1,R4    ; Initialize counter
          50   FFFFFE00  8F    D0  436A  4637  MOVL # XFFFFFE00,R0   ; Load VA into R0
                    08    50    DA  4371  4638 110$: MTPR R0,#P0BR   ; Load VA
                    51    3B    DB  4374  4639  MFPR #TB_DATA,R1   ; Read PTE into R1
                 22 51    1F    E0  4377  4640  BBS #TB_VAL,R1,130$   ; If TB hit continue test
                    55    08    DB  437B  4641  MFPR #P0BR,R5   ; Save VA
       14 0000FE00 EF    1F    E1  437E  4642  BBC #31,APT_CTRL_FLAGS,120$ ; Branch if not APT mode
       0000FE44'EF    9A AF    3E  4386  4643  MOVAW B 120$,APT_ERR_PC ; Save Error PC
          0000FE40 EF    01    88  438E  4644  BISB2 #1,APT_ERR_FLAG   ; Set Error Flag
                      C8EA    30  4395  4645  BSBW APT_COMMAND
                         03    11  4398  4646  BRB 130$
                               00  439A  4647 120$: HALT     ; ERROR,
                                   439B  4648       ; did not get TB hit at specified VA
```

```
                                    439B  4649      ; R5 contains VA
                                    439B  4650      ; R1 Bit<31> contains Valid Bit
                          D4   11   439B  4651  BRB 110$    ; Loop on error
     50   7FFF0000 8F    CA   439D  4652 130$: BICL2 #X7FFF0000,R0  ; Set up for next Index value
            08   50      DA   43A4  4653  MTPR R0,#P0BR  ; Load VA
     3B   80000000 8F    DA   43A7  4654  MTPR #X80000000,#TB_DATA ; Write PTE=80000000 into TB
                C674     30   43AE  4655  BSBW DEC_INDEX  ; Go get another Index value
     50   7FFF0000 8F    C8   43B1  4656  BISL2 #X7FFF0000,R0  ; Set VA=all ones
B1   54   00000100 8F    F3   43B8  4657  AOBLEQ #X100,R4,110$  ; Continue for 256 Index locations
```

```
                                43C0  4658  .SBTTL  TEST 27: TB DYNAMIC TEST TAG STORE MEMORY GROUP 1
                                43C0  4659 ;**
                                43C0  4660 ; FUNCTIONAL DESCRIPTION:
                                43C0  4661 ;
                                43C0  4662 ; This test verifies the ability to write and read all of the
                                43C0  4663 ; locations in the Tag Store Rams of TB Group 1 by use of a
                                43C0  4664 ; marching pattern applied to the VA Tag Store.
                                43C0  4665 ;
                                43C0  4666 ; The test consists of two subtests:
                                43C0  4667 ; Subtest 1 checks all locations while incrementing through the Index.
                                43C0  4668 ; Subtest 2 checks all locatiosn while decrementing through the Index.
                                43C0  4669 ;
                                43C0  4670 ;--
                                43C0  4671
                                43C0  4672
                                43C0  4673  SEQ  $TN    ; Macro call for Test Sequence Check
                                43C0       TEST_27::     ; Continued on next line
          0000FE50 EF    1B    B0  43C0        MOVW #27,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  43C7        INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  43CD        CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        43D3         APT_TEST_COUNT  ; Check for the correct test number
                          20   13  43D8        BEQL T27_S1    ; Sequence correct, continue test
          0000FE00 EF    1F   E1  43DA        BBC #31,APT_CTRL_FLAGS,-
                          17       43E1         T27_HLT    ; Branch if not APT mode
     0000FE44 EF    000043F9 EF   9E  43E2        MOVAB T27_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01   B0  43ED        MOVW #1,APT_ERR_FLAG  ; Set the error flag
                        C88B   30  43F4        BSBW APT_COMMAND  ; Call APT routine
                          01   11  43F7        BRB T27_S1    ; Continue
                               43F9       T27_HLT:
                          00   43F9        HALT     ; ERROR!!! Program out of sequence
                               43FA       T27_S1:
       C607 CF    FD AF    DE  43FA        MOVAL T27_S1,RET_PC  ; Store return PC
                               4400
               26    0F    DA  4400  4674  MTPR # XF,#MCESR  ; Clear MCESR
               39    00    DA  4403  4675  MTPR #0,#TBIA  ; Flush the TB
               24    0D    DA  4406  4676  MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
                               4409  4677
                               4409  4678 ;
                               4409  4679 ; The following code fills all of the TB Tag Store locations with zeroes.
                               4409  4680 ;
                               4409  4681
               54    01    D0  4409  4682  MOVL #1,R4    ; Initialize counter
                     50    D4  440C  4683  CLRL R0    ; Load initial Index value
               08    50    DA  440E  4684  MTPR R0,#P0BR  ; Load VA
          3B   80000000 8F    DA  4411  4685 10$: MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                     C5F4   30  4418  4686  BSBW INC_INDEX  ; Go get the next Index value
       EE 54   00000100 8F    F3  441B  4687  AOBLEQ # X100,R4,10$  ; Continue for 256 locations
                               4423  4688
                               4423  4689 ;
                               4423  4690 ; The next section of code reads and checks each Tag Store location
                               4423  4691 ; for VA=0, and then writes all ones into that location.
                               4423  4692 ;
                               4423  4693
               54    01    D0  4423  4694  MOVL #1,R4    ; Initialize counter
                     50    D4  4426  4695  CLRL R0    ; Load initial Index value
               08    50    DA  4428  4696  MTPR R0,#P0BR  ; Load VA
               51    3B    DB  442B  4697 20$: MFPR #TB_DATA,R1  ; Read PTE into R1
```

```
                22 51    1F    E0  442E  4698  BBS #TB_VAL,R1,40$  ; If TB hit, continue test
                   55    08    DB  4432  4699  MFPR #P0BR,R5  ; Save VA
          14 0000FE00 EF    1F    E1  4435  4700  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
          0000FE44 EF   51'AF    3E  443D  4701  MOVAW B 30$,APT_ERR_PC ; Save Error PC
             0000FE40'EF    01    88  4445  4702  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       C833    30  444C  4703  BSBW APT_COMMAND
                         03    11  444F  4704  BRB 40$
                               00  4451  4705 30$: HALT    ; ERROR,
                                   4452  4706       ; did not get TB hit at specified VA
                                   4452  4707       ; R5 contains VA
                                   4452  4708       ; R1 Bit<31> contains Valid Bit
                         D7    11  4452  4709  BRB 20$   ; Loop on error
             50    7FFF0000 8F    C8  4454  4710 40$: BISL2 # X7FFF0000,R0  ; Make VA=all ones
                      08    50    DA  445B  4711  MTPR R0,#P0BR  ; Load VA
             3B    80000000 8F    DA  445E  4712  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
             50    7FFF0000 8F    CA  4465  4713  BICL2 # X7FFF0000,R0  ; Set up to get next Index value
                       C5A0    30  446C  4714  BSBW INC_INDEX  ; Go get another Index value
          B4 54    00000100 8F    F3  446F  4715  AOBLEQ # X100,R4,20$  ; Continue for 256 locations
                                   4477  4716
                                   4477  4717 ;
                                   4477  4718 ; The next section of code reads and checks each Tag Store location
                                   4477  4719 ; for VA=ones, and then writes all zeroes into that location.
                                   4477  4720 ;
                                   4477  4721
                      54    01    D0  4477  4722  MOVL #1,R4   ; Initialize counter
             50    7FFF0000 8F    D0  447A  4723  MOVL # X7FFF0000,R0  ; Load VA into R0
                      08    50    DA  4481  4724 50$: MTPR R0,#P0BR  ; Load VA
                      51    3B    DB  4484  4725  MFPR #TB_DATA,R1  ; Read PTE into R1
                   22 51    1F    E0  4487  4726  BBS #TB_VAL,R1,70$  ; If TB hit, continue test
                      55    08    DB  448B  4727  MFPR #P0BR,R5  ; Save VA
          14 0000FE00'EF    1F    E1  448E  4728  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
          0000FE44'EF    AA AF    3E  4496  4729  MOVAW B 60$,APT_ERR_PC ; Save Error PC
             0000FE40'EF    01    88  449E  4730  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       C7DA    30  44A5  4731  BSBW APT_COMMAND
                         03    11  44A8  4732  BRB 70$
                               00  44AA  4733 60$: HALT    ; ERROR,
                                   44AB  4734       ; did not get TB hit at specified VA
                                   44AB  4735       ; R5 contains VA
                                   44AB  4736       ; R1 Bit<31> contains Valid Bit
                         D4    11  44AB  4737  BRB 50$   ; Loop on error
             50    7FFF0000 8F    CA  44AD  4738 70$: BICL2 # X7FFF0000,R0  ; Set up for next Index value
                      08    50    DA  44B4  4739  MTPR R0,#P0BR  ; Load VA
             3B    80000000 8F    DA  44B7  4740  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                       C54E    30  44BE  4741  BSBW INC_INDEX  ; Go get next Index value
             50    7FFF0000 8F    C8  44C1  4742  BISL2 # X7FFF0000,R0  ; Set VA=all ones
          B1 54    00000100 8F    F3  44C8  4743  AOBLEQ # X100,R4,50$  ; Continue for 256 locations
                         1A    11  44D0  4744  BRB T27_S2   ; Go to next subtest
                                   44D2  4745
                                   44D2  4746 ;
                                   44D2  4747 ; The next section of code is used only to initialize conditons to
                                   44D2  4748 ; execute T27_S2 in event a scope loop is desired as a result of a
                                   44D2  4749 ; Machine Check Exception. This code shortens the scope loop.
                                   44D2  4750 ;
                                   44D2  4751
                                   44D2  4752 T27_INIT1:    ; Return label for T27_S2 loop on err.
                      54    01    D0  44D2  4753  MOVL #1,R4   ; Initialize counter
                            50    D4  44D5  4754  CLRL R0   ; Load initial Index value
```

```
                    08   00    DA  44D7  4755  MTPR #0,#POBR  ; Load VA
          3B   80000000 8F    DA  44DA  4756 10$: MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                     C52B   30  44E1  4757  BSBW INC_INDEX  ; Go get the next Index value
       EE 54   00000100 8F    F3  44E4  4758  AOBLEQ # X100,R4,10$  ; Continue for 256 locations
                                   44EC  4759
                                   44EC  4760
                                   44EC  4761 ;
                                   44EC  4762 ; The next section of code reads and checks each Tag Store location
                                   44EC  4763 ; starting from the highest Index location for VA=zeroes and then
                                   44EC  4764 ; writes all ones into that location.
                                   44EC  4765 ;
                                   44EC  4766
        C515 CF    E3 AF    DE  44EC  4767 T27_S2: MOVAL T27_INIT1,RET_PC ; Store RET PC for T27_S2 loop on err.
                  54   01    D0  44F2  4768  MOVL #1,R4   ; Initialize counter
          50   8000FE00 8F    D0  44F5  4769  MOVL # X8000FE00,R0  ; Load VA into R0
                    08   50    DA  44FC  4770  MTPR R0,#POBR  ; Load VA
                    51   3B    DB  44FF  4771 80$: MFPR #TB_DATA,R1  ; Read PTE into R1
                 22 51   1F    E0  4502  4772  BBS #TB_VAL,R1,100$  ; If TB hit continue test
                    55   08    DB  4506  4773  MFPR #POBR,R5  ; Save VA
       14 0000FE00 EF   1F    E1  4509  4774  BBC #31,APT_CTRL_FLAGS,90$ ; Branch if not APT mode
       0000FE44'EF   25'AF    3E  4511  4775  MOVAW B 90$,APT_ERR_PC ; Save Error PC
          0000FE40 EF   01    88  4519  4776  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       C75F   30  4520  4777  BSBW APT_COMMAND
                         03   11  4523  4778  BRB 100$
                              00  4525  4779 90$: HALT    ; ERROR,
                                   4526  4780       ; did not get TB hit at specified VA
                                   4526  4781       ; R5 contains VA
                                   4526  4782       ; R1 Bit<31> contains Valid Bit
                       D7   11     4526  4783  BRB 80$   ; Loop on error
          50   7FFF0000 8F    C8  4528  4784 100$: BISL2 # X7FFF0000,R0  ; Make VA=all ones
                    08   50    DA  452F  4785  MTPR R0,#POBR  ; Load VA
          3B   80000000 8F    DA  4532  4786  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
          50   7FFF0000 8F    CA  4539  4787  BICL2 # X7FFF0000,R0  ; Set up for next Index value
                       C4E2   30  4540  4788  BSBW DEC_INDEX  ; Go get another Index value
       B4 54   00000100 8F    F3  4543  4789  AOBLEQ # X100,R4,80$  ; Continue for 256 Index locations
                                   454B  4790
                                   454B  4791 ;
                                   454B  4792 ; The next section for code reads and checks each Tag Store location
                                   454B  4793 ; starting at the highest Index location for a VA-ones and then
                                   454B  4794 ; writes a VA=zero into that location.
                                   454B  4795 ;
                                   454B  4796
                    54   01    D0  454B  4797  MOVL #1,R4   ; Initialize counter
          50   FFFFFE00 8F    D0  454E  4798  MOVL # XFFFFFE00,R0  ; Load VA into R0
                    08   50    DA  4555  4799 110$: MTPR R0,#POBR  ; Load VA
                    51   3B    DB  4558  4800  MFPR #TB_DATA,R1  ; Read PTE into R1
                 22 51   1F    E0  455B  4801  BBS #TB_VAL,R1,130$  ; If TB hit continue test
                    55   08    DB  455F  4802  MFPR #POBR,R5  ; Save VA
       14 0000FE00'EF   1F    E1  4562  4803  BBC #31,APT_CTRL_FLAGS,120$ ; Branch if not APT mode
       0000FE44'EF   7E'AF    3E  456A  4804  MOVAW B 120$,APT_ERR_PC ; Save Error PC
          0000FE40'EF   01    88  4572  4805  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       C706   30  4579  4806  BSBW APT_COMMAND
                         03   11  457C  4807  BRB 130$
                              00  457E  4808 120$: HALT    ; ERROR,
                                   457F  4809       ; did not get TB hit at specified VA
                                   457F  4810       ; R5 contains VA
                                   457F  4811       ; R1 Bit<31> contains Valid Bit
```

```
                        D4   11   457F   4812  BRB 110$    ; Loop on error
          50   7FFF0000 8F   CA   4581   4813 130$: BICL2 # X7FFF0000,R0  ; Set up for next Index value
               08   50   DA   4588   4814  MTPR R0,#P0BR  ; Load VA
          3B   80000000 8F   DA   458B   4815  MTPR # X80000000,#TB_DATA ; Write PTE=80000000 into TB
                     C490   30   4592   4816  BSBW DEC_INDEX  ; Go get another Index value
          50   7FFF0000 8F   C8   4595   4817  BISL2 # X7FFF0000,R0  ; Set VA=all ones
       B1 54   00000100 8F   F3   459C   4818  AOBLEQ # X100,R4,110$  ; Continue for 256 Index locations
```

```
                                45A4  4819  .SBTTL  TEST 28: TB DYNAMIC TEST DATA STORE MEMORY GROUP 0
                                45A4  4820 ;++
                                45A4  4821 ; FUNCTIONAL DESCRIPTION:
                                45A4  4822 ;
                                45A4  4823 ; This test verifies the ability to write and read all of the
                                45A4  4824 ; locations in the TB Data Store Rams of Group 0 by use of
                                45A4  4825 ; marching pattern applied to the PTE.
                                45A4  4826 ;
                                45A4  4827 ; The test consists of two subtests:
                                45A4  4828 ; Subtest 1 checks all locations while incrementing through the Index.
                                45A4  4829 ; Subtest 2 checks all locations while decrementing through the Index.
                                45A4  4830 ;
                                45A4  4831 ;--
                                45A4  4832
                                45A4  4833
                                45A4  4834  SEQ  $TN    ; Macro call for Test Sequence Check
                                45A4       TEST_28::     ; Continued on next line
          0000FE50 EF    1C    B0  45A4         MOVW #28,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  45AB         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  45B1         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        45B7          APT_TEST_COUNT  ; Check for the correct test number
                         20    13  45BC         BEQL T28_S1    ; Sequence correct, continue test
          0000FE00'EF    1F    E1  45BE         BBC #31,APT_CTRL_FLAGS,-
                         17        45C5          T28_HLT    ; Branch if not APT mode
     0000FE44 EF    000045DD'EF    9E  45C6         MOVAB T28_HLT,APT_ERR_PC ; Save the error PC
          0000FE40'EF    01    B0  45D1         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                       C6A7    30  45D8         BSBW APT_COMMAND  ; Call APT routine
                         01    11  45DB         BRB T28_S1    ; Continue
                             45DD       T28_HLT:
                             00  45DD         HALT     ; ERROR!!! Program out of sequence
                             45DE       T28_S1:
       C423 CF    FD AF    DE  45DE         MOVAL T28_S1,RET_PC  ; Store return PC
                             45E4
                   26    0F  DA  45E4  4835  MTPR # XF,#MCESR  ; Clear MCESR
                   39    00  DA  45E7  4836  MTPR #0,#TBIA  ; Flush the TB
                   24    0A  DA  45EA  4837  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                             45ED  4838
                             45ED  4839 ;
                             45ED  4840 ; The following code fills all of the TB Data Store locations with zeroes.
                             45ED  4841 ;
                             45ED  4842
                   54    01  D0  45ED  4843  MOVL #1,R4    ; Initialize counter
                         50  D4  45F0  4844  CLRL R0    ; Load VA into R0
                   08    50  DA  45F2  4845  MTPR R0,#P0BR  ; Load VA
        51    80000000 8F    D0  45F5  4846 10$: MOVL # X80000000,R1  ; Load PTE into R1
                   3B    51  DA  45FC  4847  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                      C40D    30  45FF  4848  BSBW INC_INDEX  ; Go get the next Index value
     EB 54    00000100 8F    F3  4602  4849  AOBLEQ # X100,R4,10$  ; Continue for 256 locations
                             460A  4850
                             460A  4851 ;
                             460A  4852 ; The next section of code reads and checks each TB Data Store location
                             460A  4853 ; for a PTE=80000000 and then writes a PTE=FC007FFF into that location.
                             460A  4854 ;
                             460A  4855
                   54    01  D0  460A  4856  MOVL #1,R4    ; Initialize counter
                         50  D4  460D  4857  CLRL R0    ; Load VA into R0
                   08    50  DA  460F  4858  MTPR R0,#P0BR  ; Load VA
```

```
                51   80000000 8F    D0  4612  4859 20$: MOVL #X80000000,R1  ; Load PTE into R1
                          52   3B    DB  4619  4860  MFPR #TB_DATA,R2  ; Read PTE into R2
                52   03FF8000 8F    CA  461C  4861  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
                     53   52   51    CD  4623  4862  XORL3 R1,R2,R3  ; R2 Contains expected PTE
                               22    13  4627  4863  BEQL 40$   ; Continue test is PTE correct
                          55   08    DB  4629  4864  MFPR #P0BR,R5  ; Save VA
           14 0000FE00'EF    1F    E1  462C  4865  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
           0000FE44 EF   48 AF    3E  4634  4866  MOVAW B 30$,APT_ERR_PC ; Save Error PC
              0000FE40'EF   01    88  463C  4867  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         C63C    30  4643  4868  BSBW APT_COMMAND
                           03    11  4646  4869  BRB 40$
                                 00  4648  4870 30$: HALT    ; ERROR,PTE incorrect
                                     4649  4871      ; R5 contains VA
                                     4649  4872      ; R1 contains PTE written
                                     4649  4873      ; R2 contains PTE read
                                     4649  4874      ; R3 Contains bad PTE bit positions
                           C7    11  4649  4875  BRB 20$   ; Loop on error
                51   FC007FFF 8F    D0  464B  4876 40$: MOVL #XFC007FFF,R1  ; Load PTE into R1
                          3B   51    DA  4652  4877  MTPR R1,#TB_DATA  ; Write PTE=FC007FFF into TB
                         C3B7    30  4655  4878  BSBW INC_INDEX  ; Go get the next Index value
           B2 54   00000100 8F    F3  4658  4879  AOBLEQ #X100,R4,20$  ; Continue for 256 locations
                                     4660  4880
                                     4660  4881 ;
                                     4660  4882 ; The next section of code reads and checks each Data Store location
                                     4660  4883 ; for a PTE=FC007FFF and then writes a PTE=80000000 into that location.
                                     4660  4884 ;
                                     4660  4885
                     54   01    D0  4660  4886  MOVL #1,R4   ; Initialize counter
                          50    D4  4663  4887  CLRL R0   ; Load VA into R0
                          08 50    DA  4665  4888  MTPR R0,#P0BR  ; Load VA
                51   FC007FFF 8F    D0  4668  4889 50$: MOVL #XFC007FFF,R1  ; Load PTE into R1
                          52   3B    DB  466F  4890  MFPR #TB_DATA,R2  ; Read PTE into R2
                52   03FF8000 8F    CA  4672  4891  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
                     53   52   51    CD  4679  4892  XORL3 R1,R2,R3  ; R2 contains expected PTE
                               22    13  467D  4893  BEQL 70$   ; Continue test if PTE correct
                          55   08    DB  467F  4894  MFPR #P0BR,R5  ; Save VA
           14 0000FE00'EF    1F    E1  4682  4895  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
           0000FE44 EF   9E'AF    3E  468A  4896  MOVAW B 60$,APT_ERR_PC ; Save Error PC
              0000FE40'EF   01    88  4692  4897  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         C5E6    30  4699  4898  BSBW APT_COMMAND
                           03    11  469C  4899  BRB 70$
                                 00  469E  4900 60$: HALT    ; ERROR, PTE incorrect
                                     469F  4901      ; R5 contains VA
                                     469F  4902      ; R1 contains PTE written
                                     469F  4903      ; R2 contains PTE read
                                     469F  4904      ; R3 contains bad PTE bit positions
                           C7    11  469F  4905  BRB 50$   ; Loop on error
                51   80000000 8F    D0  46A1  4906 70$: MOVL #X80000000,R1  ; Load PTE into R1
                          3B   51    DA  46A8  4907  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                         C361    30  46AB  4908  BSBW INC_INDEX  ; Go get next Index value
           B2 54   00000100 8F    F3  46AE  4909  AOBLEQ #X100,R4,50$  ; Continue for 256 locations
                           1D    11  46B6  4910  BRB T28_S2   ; Go to next subtest
                                     46B8  4911 ;
                                     46B8  4912 ; The next section of code is used only to initialize conditions to
                                     46B8  4913 ; execute T28_S2 in event a scope loop is desired as a result of a
                                     46B8  4914 ; Machine Check Exception. This code shortens the scope loop.
                                     46B8  4915 ;
```

```
                                    46B8  4916
                                    46B8  4917 T28_INIT1:    ; Return label for T28_S2 loop on err.
                54    01    D0    46B8  4918  MOVL #1,R4    ; Initialize counter
                      50    D4    46BB  4919  CLRL R0    ; Load VA into R0
                08    50    DA    46BD  4920  MTPR R0,#P0BR  ; Load VA
          51    80000000 8F    D0    46C0  4921 10$: MOVL # X80000000,R1  ; Load PTE into R1
                3B    51    DA    46C7  4922  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                   C342    30    46CA  4923  BSBW INC_INDEX  ; Go get the next Index value
       EB 54    00000100 8F    F3    46CD  4924  AOBLEQ # X100,R4,10$  ; Continue for 256 locations
                                    46D5  4925
                                    46D5  4926 ;
                                    46D5  4927 ; The next section of code reads and checks each TB Data Store location
                                    46D5  4928 ; starting at the highest Index location for a PTE=all zeroes and then
                                    46D5  4929 ; writes all ones into that location.
                                    46D5  4930 ;
                                    46D5  4931
        C32C CF    E0 AF    DE    46D5  4932 T28_S2: MOVAL T28_INIT1,RET_PC ; Store RET PC for T28_S2 loop on err.
                54    01    D0    46DB  4933  MOVL #1,R4    ; Initialize counter
          50    8000FE00 8F    D0    46DE  4934  MOVL # X8000FE00,R0  ; Load VA into R0
                08    50    DA    46E5  4935  MTPR R0,#P0BR  ; Load VA
          51    80000000 8F    D0    46E8  4936 80$: MOVL # X80000000,R1  ; Load PTE into R1
                52    3B    DB    46EF  4937  MFPR #TB_DATA,R2  ; Read PTE into R2
          52    03FF8000 8F    CA    46F2  4938  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
             53   52    51    CD    46F9  4939  XORL3 R1,R2,R3  ; R2 contains expected PTE
                      22    13    46FD  4940  BEQL 100$   ; Continue test if PTE correct
                55    08    DB    46FF  4941  MFPR #P0BR,R5  ; Save VA
       14 0000FE00 EF    1F    E1    4702  4942  BBC #31,APT_CTRL_FLAGS,90$ ; Branch if not APT mode
       0000FE44'EF    1E AF    3E    470A  4943  MOVAW B 90$,APT_ERR_PC ; Save Error PC
          0000FE40 EF    01    88    4712  4944  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                   C566    30    4719  4945  BSBW APT_COMMAND
                      03    11    471C  4946  BRB 100$
                            00    471E  4947 90$: HALT    ; ERROR, PTE incorrect
                                  471F  4948      ; R5 contains VA
                                  471F  4949      ; R1 contains PTE written
                                  471F  4950      ; R2 contains PTE read
                                  471F  4951      ; R3 contains bad PTE bit positions
                      C7    11    471F  4952  BRB 80$    ; Loop on error
          51    FC007FFF 8F    D0    4721  4953 100$: MOVL # XFC007FFF,R1  ; Load PTE into R1
                3B    51    DA    4728  4954  MTPR R1,#TB_DATA  ; Write PTE=FC007FFF into TB
                   C2F7    30    472B  4955  BSBW DEC_INDEX  ; Go get next Index value
       B2 54    00000100 8F    F3    472E  4956  AOBLEQ # X100,R4,80$  ; Continue for 256 locations
                                  4736  4957
                                  4736  4958 ;
                                  4736  4959 ; The next section of code reads and checks each TB Data Store location
                                  4736  4960 ; starting at the highest Index value for a PTE=FC007FFF and then writes
                                  4736  4961 ; a PTE=80000000 into that location.
                                  4736  4962 ;
                                  4736  4963
                54    01    D0    4736  4964  MOVL #1,R4    ; Initialize counter
          50    8000FE00 8F    D0    4739  4965  MOVL # X8000FE00,R0  ; Load VA into R0
                08    50    DA    4740  4966  MTPR R0,#P0BR  ; Load VA
          51    FC007FFF 8F    D0    4743  4967 110$: MOVL # XFC007FFF,R1  ; Load PTE into R1
                52    3B    DB    474A  4968  MFPR #TB_DATA,R2  ; Read PTE into R2
          52    03FF8000 8F    CA    474D  4969  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
             53   52    51    CD    4754  4970  XORL3 R1,R2,R3  ; R2 contains expected PTE
                      22    13    4758  4971  BEQL 130$   ; Continue test if PTE correct
                55    08    DB    475A  4972  MFPR #P0BR,R5  ; Save VA
```

```
                14 0000FE00'EF    1F    E1  475D  4973  BBC #31,APT_CTRL_FLAGS,120$ ; Branch if not APT mode
                0000FE44 EF    79'AF    3E  4765  4974  MOVAW B 120$,APT_ERR_PC ; Save Error PC
                   0000FE40 EF    01    88  476D  4975  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                                 C50B    30  4774  4976  BSBW APT_COMMAND
                                   03    11  4777  4977  BRB 130$
                                         00  4779  4978 120$: HALT    ; ERROR, PTE incorrect
                                             477A  4979       ; R5 contains VA
                                             477A  4980       ; R1 contains PTE written
                                             477A  4981       ; R2 contains PTE read
                                             477A  4982       ; R3 contains bad PTE bit positions
                                   C7    11  477A  4983  BRB 110$   ; Loop on error
                   51    80000000 8F    D0  477C  4984 130$: MOVL # X80000000,R1  ; Load PTE into R1
                            3B   51    DA  4783  4985  MTPR R1,#TB_DATA  ; Write PTE=8000000 into TB
                                 C29C    30  4786  4986  BSBW DEC_INDEX  ; Go get the next Index value
                B2 54    00000100 8F    F3  4789  4987  AOBLEQ # X100,R4,110$  ; Continue for 256 locations
```

```
                        4791  4988  .SBTTL  TEST 29: TB DYNAMIC TEST DATA STORE MEMORY GROUP 1
                        4791  4989 ;++
                        4791  4990 ; FUNCTIONAL DESCRIPTION:
                        4791  4991 ;
                        4791  4992 ; This test verifies the ability to write and read all of the
                        4791  4993 ; locations in the TB Data Store Rams of Group 1 by use of a
                        4791  4994 ; marching pattern applied to the PTE.
                        4791  4995 ;
                        4791  4996 ; The test consists of two subtests:
                        4791  4997 ; Subtest 1 checks all locations while incrementing through the Index.
                        4791  4998 ; Subtest 2 checks all locations while decrementing through the Index.
                        4791  4999 ;
                        4791  5000 ;
                        4791  5001 ;--
                        4791  5002
                        4791  5003
                        4791  5004  SEQ  $TN    ; Macro call for Test Sequence Check
                        4791        TEST_29::     ; Continued on next line
          0000FE50 EF   1D   B0   4791         MOVW #29,APT_TEST_NUM ; Load the Test Number
               0000FE52 EF   B6   4798         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50 EF   B1   479E         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        47A4          APT_TEST_COUNT  ; Check for the correct test number
                         20   13   47A9         BEQL T29_S1    ; Sequence correct, continue test
          0000FE00 EF   1F   E1   47AB         BBC #31,APT_CTRL_FLAGS,-
                         17        47B2          T29_HLT    ; Branch if not APT mode
     0000FE44 EF   000047CA EF   9E   47B3         MOVAB T29_HLT,APT_ERR_PC ; Save the error PC
               0000FE4C'EF   01   B0   47BE         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                        C4BA   30   47C5         BSBW APT_COMMAND  ; Call APT routine
                          01   11   47C8         BRB T29_S1    ; Continue
                               47CA        T29_HLT:
                          00   47CA         HALT    ; ERROR!!! Program out of sequence
                               47CB        T29_S1:
           C236 CF    FD AF   DE   47CB         MOVAL T29_S1,RET_PC  ; Store return PC
                               47D1
                    26   0F   DA   47D1  5005  MTPR # XF,#MCESR  ; Clear MCESR
                    39   00   DA   47D4  5006  MTPR #0,#TBIA  ; Flush the TB
                    24   0D   DA   47D7  5007  MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
                               47DA  5008
                               47DA  5009 ;
                               47DA  5010 ; The following code fills all of the TB Data Store locations with zeroes
                               47DA  5011 ;
                               47DA  5012
                    54   01   D0   47DA  5013  MOVL #1,R4    ; Initialize counter
                         50   D4   47DD  5014  CLRL R0    ; Load VA into R0
                    08   50   DA   47DF  5015  MTPR R0,#PBR  ; Load VA
               51   80000000 8F   D0   47E2  5016 10$: MOVL # X80000000,R1  ; Load PTE into R1
                    3B   51   DA   47E9  5017  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                        C220   30   47EC  5018  BSBW INC_INDEX  ; Go get the next Index value
          EB 54    00000100 8F   F3   47EF  5019  AOBLEQ # X100,R4,10$  ; Continue for 256 locations
                               47F7  5020
                               47F7  5021 ;
                               47F7  5022 ; The next section of code reads and checks each TB Data Store location
                               47F7  5023 ; for a PTE=80000000 and then writes a PTE=FC007FFF into that location.
                               47F7  5024 ;
                               47F7  5025
                    54   01   D0   47F7  5026  MOVL #1,R4    ; Initialize counter
                         50   D4   47FA  5027  CLRL R0    ; Load VA into R0
```

```
                    08    50    DA  47FC  5028  MTPR R0,#P0BR  ; Load VA
          51    80000000 8F    D0  47FF  5029 20$: MOVL # X80000000,R1  ; Load PTE into R1
                    52    3B    DB  4806  5030  MFPR #TB_DATA,R2  ; Read PTE into R2
          52    03FF8000 8F    CA  4809  5031  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
              53    52    51    CD  4810  5032  XORL3 R1,R2,R3  ; R2 Contains expected PTE
                          22    13  4814  5033  BEQL 40$  ; Continue test is PTE correct
                    55    08    DB  4816  5034  MFPR #P0BR,R5  ; Save VA
       14 0000FE00'EF    1F    E1  4819  5035  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
       0000FE44'EF    35 AF    3E  4821  5036  MOVAW B 30$,APT_ERR_PC ; Save Error PC
          0000FE40 EF    01    88  4829  5037  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       C44F    30  4830  5038  BSBW APT_COMMAND
                          03    11  4833  5039  BRB 40$
                          00    11  4835  5040 30$: HALT  ; ERROR, PTE incorrect
                                    4836  5041  ; R5 contains VA
                                    4836  5042  ; R1 contains PTE written
                                    4836  5043  ; R2 contains PTE read
                                    4836  5044  ; R3 Contains bad PTE bit positions
                          C7    11  4836  5045  BRB 20$  ; Loop on error
          51    FC007FFF 8F    D0  4838  5046 40$: MOVL # XFC007FFF,R1  ; Load PTE into R1
                    3B    51    DA  483F  5047  MTPR R1,#TB_DATA  ; Write PTE=FC007FFF into TB
                       C1CA    30  4842  5048  BSBW INC_INDEX  ; Go get the next Index value
       B2 54    00000100 8F    F3  4845  5049  AOBLEQ # X100,R4,20$  ; Continue for 256 locations
                                    484D  5050
                                    484D  5051 ;
                                    484D  5052 ; The next section of code reads and checks each Data Store location
                                    484D  5053 ; for a PTE=FC007FFF and then writes a PTE=80000000 into that location.
                                    484D  5054 ;
                                    484D  5055
                    54    01    D0  484D  5056  MOVL #1,R4  ; Initialize counter
                          50    D4  4850  5057  CLRL R0  ; Load VA into R0
                    08    50    DA  4852  5058  MTPR R0,#P0BR  ; Load VA
          51    FC007FFF 8F    D0  4855  5059 50$: MOVL # XFC007FFF,R1  ; Load PTE into R1
                    52    3B    DB  485C  5060  MFPR #TB_DATA,R2  ; Read PTE into R2
          52    03FF8000 8F    CA  485F  5061  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
              53    52    51    CD  4866  5062  XORL3 R1,R2,R3  ; R2 contains expected PTE
                          22    13  486A  5063  BEQL 70$  ; Continue test if PTE correct
                    55    08    DB  486C  5064  MFPR #P0BR,R5  ; Save VA
       14 0000FE00 EF    1F    E1  486F  5065  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
       0000FE44 EF    8B AF    3E  4877  5066  MOVAW B 60$,APT_ERR_PC ; Save Error PC
          0000FE40 EF    01    88  487F  5067  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       C3F9    30  4886  5068  BSBW APT_COMMAND
                          03    11  4889  5069  BRB 70$
                          00    11  488B  5070 60$: HALT  ; ERROR, PTE incorrect
                                    488C  5071  ; R5 contains VA
                                    488C  5072  ; R1 contains PTE written
                                    488C  5073  ; R2 contains PTE read
                                    488C  5074  ; R3 contains bad PTE bit positions
                          C7    11  488C  5075  BRB 50$  ; Loop on error
          51    80000000 8F    D0  488E  5076 70$: MOVL # X80000000,R1  ; Load PTE into R1
                    3B    51    DA  4895  5077  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                       C174    30  4898  5078  BSBW INC_INDEX  ; Go get next Index value
       B2 54    00000100 8F    F3  489B  5079  AOBLEQ # X100,R4,50$  ; Continue for 256 locations
                          1D    11  48A3  5080  BRB T29_S2  ; Go to next subtest
                                    48A5  5081
                                    48A5  5082 ;
                                    48A5  5083 ; The next section of code is used only to initialize conditions to
                                    48A5  5084 ; execute T29_S2 in event a scope loop is desired as a result of a
```

```
                                48A5  5085 ; Machine Check Exception. This code shortens the scope loop.
                                48A5  5086 ;
                                48A5  5087
                                48A5  5088 T29_INIT1:    ; Return label for T29_S2 loop on err.
                54    01   D0   48A5  5089  MOVL #1,R4    ; Initialize counter
                      50   D4   48A8  5090  CLRL R0    ; Load VA into R0
                08    50   DA   48AA  5091  MTPR R0,#P0BR  ; Load VA
          51    80000000 8F   D0   48AD  5092 10$: MOVL # X80000000,R1  ; Load PTE into R1
                3B    51   DA   48B4  5093  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                   C155    30   48B7  5094  BSBW INC_INDEX  ; Go get the next Index value
       EB 54    00000100 8F   F3   48BA  5095  AOBLEQ # X100,R4,10$  ; Continue for 256 locations
                                48C2  5096
                                48C2  5097 ;
                                48C2  5098 ; The next section of code reads and checks each TB Data Store location
                                48C2  5099 ; starting at the highest Index location for a PTE=all zeroes and then
                                48C2  5100 ; writes all ones into that location.
                                48C2  5101 ;
                                48C2  5102
        C13F CF    E0 AF   DE   48C2  5103 T29_S2: MOVAL T29_INIT1,RET_PC ; Store RET PC for T29_S2 loop on err
                54    01   D0   48C8  5104  MOVL #1,R4    ; Initialize counter
          50    8000FE00 8F   D0   48CB  5105  MOVL # X8000FE00,R0  ; Load VA into R0
                08    50   DA   48D2  5106  MTPR R0,#P0BR  ; Load VA
          51    80000000 8F   D0   48D5  5107 80$: MOVL # X80000000,R1  ; Load PTE into R1
                52    3B   DB   48DC  5108  MFPR #TB_DATA,R2  ; Read PTE into R2
          52    03FF8000 8F   CA   48DF  5109  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
             53   52    51   CD   48E6  5110  XORL3 R1,R2,R3  ; R2 contains expected PTE
                      22   13   48EA  5111  BEQL 100$   ; Continue test if PTE correct
                55    08   DB   48EC  5112  MFPR #P0BR,R5  ; Save VA
       14 0000FE00 EF    1F   E1   48EF  5113  BBC #31,APT_CTRL_FLAGS,90$ ; Branch if not APT mode
       0000FE44 EF    0B AF   3E   48F7  5114  MOVAW B 90$,APT_ERR_PC ; Save Error PC
          0000FE40'EF    01   88   48FF  5115  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                   C379    30   4906  5116  BSBW APT_COMMAND
                      03   11   4909  5117  BRB 100$
                           00   490B  5118 90$: HALT     ; ERROR, PTE incorrect
                                490C  5119       ; R5 contains VA
                                490C  5120       ; R1 contains PTE written
                                490C  5121       ; R2 contains PTE read
                                490C  5122       ; R3 contains bad PTE bit positions
                      C7   11   490C  5123  BRB 80$    ; Loop on error
          51    FC007FFF 8F   D0   490E  5124 100$: MOVL # XFC007FFF,R1  ; Load PTE into R1
                3B    51   DA   4915  5125  MTPR R1,#TB_DATA  ; Write PTE=FC007FFF into TB
                   C10A    30   4918  5126  BSBW DEC_INDEX  ; Go get next Index value
       B2 54    00000100 8F   F3   491B  5127  AOBLEQ # X100,R4,80$  ; Continue for 256 locations
                                4923  5128
                                4923  5129 ;
                                4923  5130 ; The next section of code reads and checks each TB Data Store location
                                4923  5131 ; starting at the highest Index value for a PTE=FC007FFF and then writes
                                4923  5132 ; a PTE=80000000 into that location.
                                4923  5133 ;
                                4923  5134
                54    01   D0   4923  5135  MOVL #1,R4    ; Initialize counter
          50    8000FE00 8F   D0   4926  5136  MOVL # X8000FE00,R0  ; Load VA into R0
                08    50   DA   492D  5137  MTPR R0,#P0BR  ; Load VA
          51    FC007FFF 8F   D0   4930  5138 110$: MOVL # XFC007FFF,R1  ; Load PTE into R1
                52    3B   DB   4937  5139  MFPR #TB_DATA,R2  ; Read PTE into R2
          52    03FF8000 8F   CA   493A  5140  BICL2 #PTE_MASK,R2  ; Clear PTE<25:15>, (indeterminate)
             53   52    51   CD   4941  5141  XORL3 R1,R2,R3  ; R2 contains expected PTE
```

```
                    22   13  4945  5142  BEQL 130$   ; Continue test if PTE correct
                 55 08   DB  4947  5143  MFPR #P0BR,R5  ; Save VA
      14 0000FE00'EF 1F   E1  494A  5144  BBC #31,APT_CTRL_FLAGS,120$ ; Branch if not APT mode
      0000FE44'EF  66'AF  3E  4952  5145  MOVAW B 120$,APT_ERR_PC ; Save Error PC
         0000FE40'EF 01   88  495A  5146  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                  C31E   30  4961  5147  BSBW APT_COMMAND
                    03   11  4964  5148  BRB 130$
                         00  4966  5149 120$: HALT    ; ERROR, PTE incorrect
                             4967  5150       ; R5 contains VA
                             4967  5151       ; R1 contains PTE written
                             4967  5152       ; R2 contains PTE read
                             4967  5153       ; R3 contains bad PTE bit positions
                    C7   11  4967  5154  BRB 110$   ; Loop on error
         51 80000000 8F   D0  4969  5155 130$: MOVL # X80000000,R1  ; Load PTE into R1
                 3B 51   DA  4970  5156  MTPR R1,#TB_DATA  ; Write PTE=8000000 into TB
                  C0AF   30  4973  5157  BSBW DEC_INDEX  ; Go get the next Index value
      B2 54 00000100 8F   F3  4976  5158  AOBLEQ # X100,R4,110$  ; Continue for 256 locations
```

```
                                497E  5159  .SBTTL  TEST 30: TB DYNAMIC TEST OF VALID BIT GROUP 0
                                497E  5160
                                497E  5161 ;++
                                497E  5162 ; FUNCTIONAL DESCRIPTION:
                                497E  5163 ;
                                497E  5164 ; This test verifies the ability to write and read all of
                                497E  5165 ; Valid Bit locations in the TB Tag Store of Group 0 by
                                497E  5166 ; use of a marching pattern applied to the Valid Bit.
                                497E  5167 ;
                                497E  5168 ; The test consists of two subtests:
                                497E  5169 ; Subtest 1 checks all locations while incrementing through the Index.
                                497E  5170 ; Subtest 2 checks all locations while decrementing through the Index.
                                497E  5171 ;
                                497E  5172 ;--
                                497E  5173
                                497E  5174
                                497E  5175  SEQ  $TN    ; Macro call for Test Sequence Check
                                497E        TEST_30::     ; Continued on next line
          0000FE50'EF    1E    B0  497E         MOVW #30,APT_TEST_NUM ; Load the Test Number
               0000FE52 EF    B6  4985         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  498B         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        4991          APT_TEST_COUNT  ; Check for the correct test number
                     20    13  4996         BEQL T30_S1    ; Sequence correct, continue test
          0000FE00'EF    1F  E1  4998         BBC #31,APT_CTRL_FLAGS,-
                     17        499F          T30_HLT    ; Branch if not APT mode
     0000FE44 EF   000049B7'EF  9E  49A0         MOVAB T30_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF    01    B0  49AB         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                    C2CD    30  49B2         BSBW APT_COMMAND  ; Call APT routine
                      01    11  49B5         BRB T30_S1    ; Continue
                                49B7        T30_HLT:
                                00  49B7         HALT    ; ERROR!!! Program out of sequence
                                49B8        T30_S1:
       C049 CF    FD AF    DE  49B8         MOVAL T30_S1,RET_PC  ; Store return PC
                                49BE
                 26    0F    DA  49BE  5176  MTPR # XF,#MCESR  ; Clear MCESR
                 39    00    DA  49C1  5177  MTPR #0,#TBIA  ; Flush the TB
                 24    0A    DA  49C4  5178  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                                49C7  5179
                                49C7  5180 ;
                                49C7  5181 ; The following code reads and checks that each TB Valid Bit is clear, and
                                49C7  5182 ; then sets each Valid Bit. All Valid Bits are initially cleared by virtue
                                49C7  5183 ; of the TB Flush.
                                49C7  5184 ;
                                49C7  5185
                 54    01    D0  49C7  5186  MOVL #1,R4    ; Initialize counter
                       50    D4  49CA  5187  CLRL R0    ; Load VA into R0
                 08    50    DA  49CC  5188  MTPR R0,#P0BR  ; Load VA
                 51    3B    DB  49CF  5189 10$: MFPR #TB_DATA,R1  ; Read PTE into R1
              22 51    1F    E1  49D2  5190  BBC #TB_VAL,R1,30$  ; If TB miss, continue test
                 55    08    DB  49D6  5191  MFPR #P0BR,R5  ; Save VA
     14 0000FE00 EF    1F    E1  49D9  5192  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
     0000FE44 EF     F5'AF    3E  49E1  5193  MOVAW B 20$,APT_ERR_PC ; Save Error PC
        0000FE40 EF    01    88  49E9  5194  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                    C28F    30  49F0  5195  BSBW APT_COMMAND
                      03    11  49F3  5196  BRB 30$
                                00  49F5  5197 20$: HALT    ; ERROR, did not get TB miss at spec VA
                                49F6  5198       ; R5 contains VA
```

```
                                 49F6  5199       ; R1 Bit<31> contains Valid Bit
                       D7   11   49F6  5200  BRB 10$    ; Loop on error
          51  80000000 8F   D0   49F8  5201 30$: MOVL # X80000000,R1  ; Load PTE into R1
              3B   51   DA   49FF  5202  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                 C00A   30   4A02  5203  BSBW INC_INDEX  ; Go get the next Index value
       C2 54  00000100 8F   F3   4A05  5204  AOBLEQ # X100,R4,10$  ; Continue for 256 locations
                                 4A0D  5205
                                 4A0D  5206 ;
                                 4A0D  5207 ; The next section of code reads and checks that each TB Valid Bit is set,
                                 4A0D  5208 ; and then clears the Valid bit.
                                 4A0D  5209 ;
                                 4A0D  5210
              54   01   D0   4A0D  5211  MOVL #1,R4    ; Initialize counter
                   50   D4   4A10  5212  CLRL R0    ; Load VA into R0
              08   50   DA   4A12  5213  MTPR R0,#P0BR  ; Load VA
              51   3B   DB   4A15  5214 40$: MFPR #TB_DATA,R1  ; Read PTE into R1
           22 51   1F   E0   4A18  5215  BBS #TB_VAL,R1,60$  ; If TB hit, continue test
              55   08   DB   4A1C  5216  MFPR #P0BR,R5  ; Save VA
    14 0000FE00'EF   1F   E1   4A1F  5217  BBC #31,APT_CTRL_FLAGS,50$ ; Branch if not APT mode
    0000FE44'EF  3B'AF   3E   4A27  5218  MOVAW B 50$,APT_ERR_PC ; Save Error PC
       0000FE40'EF   01   88   4A2F  5219  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                C249   30   4A36  5220  BSBW APT_COMMAND
                   03   11   4A39  5221  BRB 60$
                        00   4A3B  5222 50$: HALT    ; ERROR, did not get TB hit at spec. VA
                                 4A3C  5223       ; R5 contains VA
                                 4A3C  5224       ; R1 Bit<31> contains Valid Bit
                       D7   11   4A3C  5225  BRB 40$    ; Loop on error
                   51   D4   4A3E  5226 60$: CLRL R1    ; Load PTE into R1
              3B   51   DA   4A40  5227  MTPR R1,#TB_DATA  ; Write PTE=00000000 into TB
                 BFC9   30   4A43  5228  BSBW INC_INDEX  ; Go get the next Index value
       C7 54  00000100 8F   F3   4A46  5229  AOBLEQ # X100,R4,40$  ; Continue for 256 locations
                   03   11   4A4E  5230  BRB T30_S2    ; Go to next subtest
                                 4A50  5231
                                 4A50  5232 ;
                                 4A50  5233 ; The next instruction is used only to initialize conditions to
                                 4A50  5234 ; execute T30_S2 in event a scope loop is desired as a result of a
                                 4A50  5235 ; Machine Check Exception. This instruction shortens the scope loop.
                                 4A50  5236 ;
                                 4A50  5237
                                 4A50  5238 T30_INIT1:     ; Return label for T30_S2 loop on err.
              39   00   DA   4A50  5239  MTPR #0,#TBIA  ; Flush the TB
                                 4A53  5240
                                 4A53  5241
                                 4A53  5242 ;
                                 4A53  5243 ; The next section of code reads and checks that each TB Valid Bit is clear,
                                 4A53  5244 ; starting at the highest Index location and then sets the Valid Bit
                                 4A53  5245 ;
                                 4A53  5246
        BFAE CF    FA AF   DE   4A53  5247 T30_S2: MOVAL T30_INIT1,RET_PC ; Store RET PC for T30_S2 loop on err.
              54   01   D0   4A59  5248  MOVL #1,R4    ; Initialize counter
       50  8000FE00 8F   D0   4A5C  5249  MOVL # X8000FE00,R0  ; Load VA into R0
              08   50   DA   4A63  5250  MTPR R0,#P0BR  ; Load VA
              51   3B   DB   4A66  5251 70$: MFPR #TB_DATA,R1  ; Read PTE into R1
           22 51   1F   E1   4A69  5252  BBC #TB_VAL,R1,90$  ; If TB miss, continue test
              55   08   DB   4A6D  5253  MFPR #P0BR,R5  ; Save VA
    14 0000FE00 EF   1F   E1   4A70  5254  BBC #31,APT_CTRL_FLAGS,80$ ; Branch if not APT mode
    0000FE44 EF   8C AF   3E   4A78  5255  MOVAW B 80$,APT_ERR_PC ; Save Error PC
```

```
                0000FE40 EF    01    88  4A80  5256  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       C1F8    30    4A87  5257  BSBW APT_COMMAND
                         03    11    4A8A  5258  BRB 90$
                               00    4A8C  5259 80$: HALT    ; ERROR, did not get TB miss at spec VA
                                     4A8D  5260      ; R5 contains VA
                                     4A8D  5261      ; R1 Bit<31> contains Valid Bit
                         D7    11    4A8D  5262  BRB 70$   ; Loop on error
             51  80000000 8F    D0    4A8F  5263 90$: MOVL # X80000000,R1  ; Load PTE into R1
                     3B  51    DA    4A96  5264  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                       BF89    30    4A99  5265  BSBW DEC_INDEX  ; Go get the next Index value
          C2 54  00000100 8F    F3    4A9C  5266  AOBLEQ # X100,R4,70$  ; Continue for 256 locations
                                     4AA4  5267
                                     4AA4  5268 ;
                                     4AA4  5269 ; The next section of code reads and checks that each TB Valid Bit is set,
                                     4AA4  5270 ; starting at the highest Index location and then clears the Valid Bit.
                                     4AA4  5271 ;
                                     4AA4  5272
                     54    01    D0    4AA4  5273  MOVL #1,R4   ; Initialize counter
             50  8000FE00 8F    D0    4AA7  5274  MOVL # X8000FE00,R0  ; Load VA into R0
                     08    50    DA    4AAE  5275  MTPR R0,#P0BR  ; Load VA
                     51    3B    DB    4AB1  5276 100$: MFPR #TB_DATA,R1  ; Read PTE into R1
                   22 51    1F    E0    4AB4  5277  BBS #TB_VAL,R1,120$  ; If TB hit, continue test
                     55    08    DB    4AB8  5278  MFPR #P0BR,R5  ; Save VA
          14 0000FE00'EF    1F    E1    4ABB  5279  BBC #31,APT_CTRL_FLAGS,110$ ; Branch if not APT mode
          0000FE44 EF  D7'AF    3E    4AC3  5280  MOVAW B 110$,APT_ERR_PC ; Save Error PC
             0000FE40'EF    01    88    4ACB  5281  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       C1AD    30    4AD2  5282  BSBW APT_COMMAND
                         03    11    4AD5  5283  BRB 120$
                               00    4AD7  5284 110$: HALT    ; ERROR, did not get TB hit at spec. VA
                                     4AD8  5285      ; R5 contains VA
                                     4AD8  5286      ; R1 Bit<31> contains Valid Bit
                         D7    11    4AD8  5287  BRB 100$   ; Loop on error
                         51    D4    4ADA  5288 120$: CLRL R1   ; Load PTE into R1
                     3B  51    DA    4ADC  5289  MTPR R1,#TB_DATA  ; Write PTE=00000000 into TB
                       BF43    30    4ADF  5290  BSBW DEC_INDEX  ; Go get the next Index value
          C7 54  00000100 8F    F3    4AE2  5291  AOBLEQ # X100,R4,100$  ; Continue for 256 locations
```

```
                        4AEA  5292  .SBTTL  TEST 31: TB DYNAMIC TEST OF VALID BIT GROUP 1
                        4AEA  5293
                        4AEA  5294 ;++
                        4AEA  5295 ; FUNCTIONAL DESCRIPTION:
                        4AEA  5296 ;
                        4AEA  5297 ; This test verifies the ability to write and read all of the
                        4AEA  5298 ; Valid Bit locations in the TB Tag Store of Group 1 by use
                        4AEA  5299 ; of a marching pattern applied to the Valid Bit.
                        4AEA  5300 ;
                        4AEA  5301 ; The test consists of two subtests:
                        4AEA  5302 ; Subtest 1 checks all locations while incrementing through the Index.
                        4AEA  5303 ; Subtest 2 checks all locations while decrementing through the Index.
                        4AEA  5304 ;
                        4AEA  5305 ;
                        4AEA  5306 ;
                        4AEA  5307 ;--
                        4AEA  5308
                        4AEA  5309
                        4AEA  5310  SEQ  $TN    ; Macro call for Test Sequence Check
                        4AEA        TEST_31::     ; Continued on next line
          0000FE50 EF   1F   B0   4AEA          MOVW #31,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF   B6   4AF1          INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF   B1   4AF7          CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        4AFD           APT_TEST_COUNT  ; Check for the correct test number
                         20   13   4B02          BEQL T31_S1    ; Sequence correct, continue test
          0000FE00'EF   1F   E1   4B04          BBC #31,APT_CTRL_FLAGS,-
                         17        4B0B           T31_HLT    ; Branch if not APT mode
     0000FE44'EF   00004B23'EF   9E   4B0C          MOVAB T31_HLT,APT_ERR_PC ; Save the error PC
          0000FE40'EF   01   B0   4B17          MOVW #1,APT_ERR_FLAG  ; Set the error flag
                      C161   30   4B1E          BSBW APT_COMMAND  ; Call APT routine
                         01   11   4B21          BRB T31_S1    ; Continue
                              4B23         T31_HLT:
                         00   4B23          HALT     ; ERROR!!! Program out of sequence
                              4B24         T31_S1:
      BEDD CF    FD AF    DE   4B24          MOVAL T31_S1,RET_PC  ; Store return PC
                              4B2A
                 26   0F   DA   4B2A  5311  MTPR # XF,#MCESR  ; Clear MCESR
                 39   00   DA   4B2D  5312  MTPR #0,#TBIA  ; Flush the TB
                 24   0D   DA   4B30  5313  MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
                              4B33  5314
                              4B33  5315 ;
                              4B33  5316 ; The following code reads and checks that each TB Valid Bit is clear, and
                              4B33  5317 ; then sets each Valid Bit. All Valid Bits are initially cleared by virtue
                              4B33  5318 ; of the TB Flush.
                              4B33  5319 ;
                              4B33  5320
                 54   01   D0   4B33  5321  MOVL #1,R4    ; Initialize counter
                      50   D4   4B36  5322  CLRL R0     ; Load VA into R0
                 08   50   DA   4B38  5323  MTPR R0,#P0BR  ; Load VA
                 51   3B   DB   4B3B  5324 10$: MFPR #TB_DATA,R1  ; Read PTE into R1
               22 51   1F   E1   4B3E  5325  BBC #TB_VAL,R1,30$  ; If TB miss, continue test
                 55   08   DB   4B42  5326  MFPR #P0BR,R5  ; Save VA
     14 0000FE00'EF   1F   E1   4B45  5327  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
     0000FE44 EF    61'AF   3E   4B4D  5328  MOVAW B 20$,APT_ERR_PC ; Save Error PC
          0000FE40'EF   01   88   4B55  5329  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      C123   30   4B5C  5330  BSBW APT_COMMAND
                         03   11   4B5F  5331  BRB 30$
```

```
                                00  4B61  5332 20$: HALT     ; ERROR, did not get TB miss at spec VA
                                    4B62  5333        ; R5 contains VA
                                    4B62  5334        ; R1 Bit<31> contains Valid Bit
                             D7  11  4B62  5335  BRB 10$    ; Loop on error
          51    80000000 8F  D0  4B64  5336 30$: MOVL #^X80000000,R1  ; Load PTE into R1
                       3B  51  DA  4B6B  5337  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
                          BE9E 30  4B6E  5338  BSBW INC_INDEX  ; Go get the next Index value
       C2 54    00000100 8F  F3  4B71  5339  AOBLEQ #^X100,R4,10$  ; Continue for 256 locations
                                    4B79  5340
                                    4B79  5341 ;
                                    4B79  5342 ; The next section of code reads and checks that each TB Valid Bit is set,
                                    4B79  5343 ; and then clears the Valid bit.
                                    4B79  5344 ;
                                    4B79  5345
                       54  01  D0  4B79  5346  MOVL #1,R4    ; Initialize counter
                           50  D4  4B7C  5347  CLRL R0    ; Load VA into R0
                       08  50  DA  4B7E  5348  MTPR R0,#P0BR  ; Load VA
                       51  3B  DB  4B81  5349 40$: MFPR #TB_DATA,R1  ; Read PTE into R1
                    22 51  1F  E0  4B84  5350  BBS #TB_VAL,R1,60$  ; If TB hit, continue test
                       55  08  DB  4B88  5351  MFPR #P0BR,R5  ; Save VA
       14 0000FE00 EF  1F  E1  4B8B  5352  BBC #31,APT_CTRL_FLAGS,50$ ; Branch if not APT mode
       0000FE44'EF   A7'AF  3E  4B93  5353  MOVAW B^50$,APT_ERR_PC ; Save Error PC
          0000FE40'EF   01  88  4B9B  5354  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                          C0DD 30  4BA2  5355  BSBW APT_COMMAND
                             03  11  4BA5  5356  BRB 60$
                                 00  4BA7  5357 50$: HALT     ; ERROR, did not get TB hit at spec. VA
                                     4BA8  5358        ; R5 contains VA
                                     4BA8  5359        ; R1 Bit<31> contains Valid Bit
                             D7  11  4BA8  5360  BRB 40$    ; Loop on error
                             51  D4  4BAA  5361 60$: CLRL R1    ; Load PTE into R1
                       3B    51  DA  4BAC  5362  MTPR R1,#TB_DATA  ; Write PTE=00000000 into TB
                          BE5D 30  4BAF  5363  BSBW INC_INDEX  ; Go get the next Index value
          C7 54    00000100 8F  F3  4BB2  5364  AOBLEQ #^X100,R4,40$  ; Continue for 256 locations
                             03  11  4BBA  5365  BRB T31_S2    ; Go to next subtest
                                     4BBC  5366
                                     4BBC  5367 ;
                                     4BBC  5368 ; The next instruction is used only to initialize conditions to
                                     4BBC  5369 ; execute T31_S2 in event a scope loop is desired as a result of a
                                     4BBC  5370 ; Machine Check Exception. This instruction shortens the scope loop.
                                     4BBC  5371 ;
                                     4BBC  5372
                                     4BBC  5373 T31_INIT1:     ; Return label for T31_S2 loop on err.
                       39  00  DA  4BBC  5374  MTPR #0,#TBIA  ; Flush the TB
                                     4BBF  5375
                                     4BBF  5376
                                     4BBF  5377 ;
                                     4BBF  5378 ; The next section of code reads and checks that each TB Valid Bit is clear,
                                     4BBF  5379 ; starting at the highest Index location and then sets the Valid Bit
                                     4BBF  5380 ;
                                     4BBF  5381
          BE41 CF    FE8D CF  DE  4BBF  5382 T31_S2: MOVAL T30_INIT1,RET_PC ; Store RET PC for T31_S2 loop on err.
                       54  01  D0  4BC6  5383  MOVL #1,R4    ; Initialize counter
           50    8000FE00 8F  D0  4BC9  5384  MOVL #^X8000FE00,R0  ; Load VA into R0
                       08  50  DA  4BD0  5385  MTPR R0,#P0BR  ; Load VA
                       51  3B  DB  4BD3  5386 70$: MFPR #TB_DATA,R1  ; Read PTE into R1
                    22 51  1F  E1  4BD6  5387  BBC #TB_VAL,R1,90$  ; If TB miss, continue test
                       55  08  DB  4BDA  5388  MFPR #P0BR,R5  ; Save VA
```

```
                                                4BDD  5389  BBC #31,APT_CTRL_FLAGS,80$ ; Branch if not APT mode
14 0000FE00'EF    1F    E1                      4BE5  5390  MOVAW B 80$,APT_ERR_PC ; Save Error PC
0000FE44 EF    F9 AF    3E                      4BED  5391  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
   0000FE40 EF    01    88                      4BF4  5392  BSBW APT_COMMAND
                C08B    30                      4BF7  5393  BRB 90$
                  03    11                      4BF9  5394 80$: HALT    ; ERROR, did not get TB miss at spec VA
                                                4BFA  5395      ; R5 contains VA
                                                4BFA  5396      ; R1 Bit<31> contains Valid Bit
                  D7    11                      4BFA  5397  BRB 70$   ; Loop on error
   51    80000000 8F    D0                      4BFC  5398 90$: MOVL # X80000000,R1  ; Load PTE into R1
            3B   51    DA                      4C03  5399  MTPR R1,#TB_DATA  ; Write PTE=80000000 into TB
              BE1C    30                      4C06  5400  BSBW DEC_INDEX  ; Go get the next Index value
C2 54    00000100 8F    F3                      4C09  5401  AOBLEQ # X100,R4,70$  ; Continue for 256 locations
                                                4C11  5402
                                                4C11  5403 ;
                                                4C11  5404 ; The next section of code reads and checks that each TB Valid Bit is set,
                                                4C11  5405 ; starting at the highest Index location and then clears the Valid Bit.
                                                4C11  5406 ;
                                                4C11  5407
            54    01    D0                      4C11  5408  MOVL #1,R4   ; Initialize counter
   50    8000FE00 8F    D0                      4C14  5409  MOVL # X8000FE00,R0  ; Load VA into R0
            08    50    DA                      4C1B  5410  MTPR R0,#P0BR  ; Load VA
            51    3B    DB                      4C1E  5411 100$: MFPR #TB_DATA,R1  ; Read PTE into R1
         22 51    1F    E0                      4C21  5412  BBS #TB_VAL,R1,120$  ; If TB hit, continue test
            55    08    DB                      4C25  5413  MFPR #P0BR,R5  ; Save VA
14 0000FE00 EF    1F    E1                      4C28  5414  BBC #31,APT_CTRL_FLAGS,110$ ; Branch if not APT mode
0000FE44'EF    44'AF    3E                      4C30  5415  MOVAW B 110$,APT_ERR_PC ; Save Error PC
   0000FE40'EF    01    88                      4C38  5416  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                C040    30                      4C3F  5417  BSBW APT_COMMAND
                  03    11                      4C42  5418  BRB 120$
                                                4C44  5419 110$: HALT    ; ERROR, did not get TB hit at spec. VA
                                                4C45  5420      ; R5 contains VA
                                                4C45  5421      ; R1 Bit<31> contains Valid Bit
                  D7    11                      4C45  5422  BRB 100$   ; Loop on error
                  51    D4                      4C47  5423 120$: CLRL R1   ; Load PTE into R1
            3B    51    DA                      4C49  5424  MTPR R1,#TB_DATA  ; Write PTE=00000000 into TB
              BDD6    30                      4C4C  5425  BSBW DEC_INDEX  ; Go get the next Index value
C7 54    00000100 8F    F3                      4C4F  5426  AOBLEQ # X100,R4,100$  ; Continue for 256 locations
```

```
                                4C57  5427  .SBTTL  TEST 32: TB RANDOM GROUP REPLACEMENT TEST
                                4C57  5428 ;++
                                4C57  5429 ; FUNCTIONAL DESCRIPTION:
                                4C57  5430 ;
                                4C57  5431 ; This test checks that random replacement occurs in both groups.
                                4C57  5432 ; A TB Flush is first performed to invalidate all TB locations.
                                4C57  5433 ; Random replacement is then enabled and 2 TB locations are
                                4C57  5434 ; written with a PTE=80000000.  The two NOP's plus the MTPR instruc-
                                4C57  5435 ; tion inbetween the two MTPR Rn,#TB_DATA instructions, result in a
                                4C57  5436 ; clock-extended series to force a write to the opposing group.  i.e.
                                4C57  5437 ; (assuming no hardware faults) if Group 0 was written first, Group 1
                                4C57  5438 ; would be hit on the 2nd write, and vice versa.  Then, Group 0 is
                                4C57  5439 ; examined for a hit on the first write.  If a hit was made, a check
                                4C57  5440 ; for a hit on Group 1 on the second write is made.  If Group 0 was
                                4C57  5441 ; NOT hit, Group 1 is checked for a hit on the first write, then a
                                4C57  5442 ; hit on Group 0 is checked for the second write.  Any hit not found
                                4C57  5443 ; to the above algorithm will cause an error.
                                4C57  5444 ;
                                4C57  5445 ;
                                4C57  5446 ;--
                                4C57  5447  SEQ  $TN    ; Macro call for Test Sequence Check
                                4C57       TEST_32::    ; Continued on next line
          0000FE50'EF   20   B0 4C57        MOVW #32,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF   B6 4C5E        INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF   B1 4C64        CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52 EF      4C6A         APT_TEST_COUNT  ; Check for the correct test number
                          20   13 4C6F        BEQL T32_S1    ; Sequence correct, continue test
          0000FE00 EF   1F   E1 4C71        BBC #31,APT_CTRL_FLAGS,-
                          17      4C78         T32_HLT    ; Branch if not APT mode
     0000FE44 EF   00004C90 EF   9E 4C79        MOVAB T32_HLT,APT_ERR_PC ; Save the error PC
               0000FE40 EF   01   B0 4C84        MOVW #1,APT_ERR_FLAG  ; Set the error flag
                        BFF4   30 4C8B        BSBW APT_COMMAND  ; Call APT routine
                          01   11 4C8E        BRB T32_S1    ; Continue
                               4C90       T32_HLT:
                          00   4C90        HALT    ; ERROR!!! Program out of sequence
                               4C91       T32_S1:
           BD70 CF    FD AF   DE 4C91        MOVAL T32_S1,RET_PC  ; Store return PC
                               4C97
                          52   D4 4C97  5448  CLRL R2    ; Clear R2
          53    80000000 8F   D0 4C99  5449  MOVL # X80000000,R3  ; Load R3 with  X80000000
          54    00000200 8F   D0 4CA0  5450  MOVL # X200,R4  ; Load R4 with  X200
                     55   01   D0 4CA7  5451  MOVL #1,R5    ; Setup loop counter
                     25   00   DA 4CAA  5452  MTPR #0,#CADR  ; Ensure Cache is enabled
                     24   00   DA 4CAD  5453  MTPR #0,#TBDR  ; Enable Random Replacement
                     26   0F   DA 4CB0  5454  MTPR # XF,#MCESR  ; Clear MCESR
```

```
                                4CB3  5455 T32_S1_START:
          BD4E CF    FD AF   DE  4CB3  5456  MOVAL B T32_S1_START,RET_PC ; Store return PC
                  39    00   DA  4CB9  5457  MTPR #0,#TBIA  ; Flush the TB
                  08    52   DA  4CBC  5458  MTPR R2,#P0BR  ; Load VA
                  3B    53   DA  4CBF  5459  MTPR R3,#TB_DATA  ; Write PTE=80000000 into TB
                               01  4CC2  5460  NOP    ; Stall
                               01  4CC3  5461  NOP    ; Stall some more
                  08    54   DA  4CC4  5462  MTPR R4,#P0BR  ; Load VA
                  3B    53   DA  4CC7  5463  MTPR R3,#TB_DATA  ; Write PTE=80000000 into TB
               E5 55    02   F3  4CCA  5464  AOBLEQ #2,R5,T32_S1_START ; Do the writing exercise twice
                                4CCE  5465
                  24    0A   DA  4CCE  5466  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
                  08    00   DA  4CD1  5467  MTPR #0,#P0BR  ; Load VA
                  51    3B   DB  4CD4  5468  MFPR #TB_DATA,R1  ; Read PTE into R1
                                4CD7  5469 ; Is there a TB hit in Group 0, VA=0?
                                4CD7  5470 ;   Branch if so to check for TB hit in Group 1, VA=200(X).
             2D 51    1F   E0  4CD7  5471  BBS #TB_VAL,R1 20$  ; If TB hit, go to section 20$
          08   00000200 8F   DA  4CDB  5472  MTPR # X200,#P0BR  ; Load VA
                  51    3B   DB  4CE2  5473  MFPR #TB_DATA,R1  ; Read PTE into R1
                                4CE5  5474 ; Is there a TB hit in Group 0, VA=200(X)?
                                4CE5  5475 ;   Branch if so to check for TB hit in Group 1, VA=0.
             50 51    1F   E0  4CE5  5476  BBS #TB_VAL,R1,40$  ; If TB hit, go to section 40$
       14 0000FE00 EF    1F   E1  4CE9  5477  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT
       0000FE44 EF    05'AF   3E  4CF1  5478  MOVAW B 10$,APT_ERR_PC ; Save Error PC
          0000FE40'EF    01   88  4CF9  5479  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      BF7F   30  4D00  5480  BSBW APT_COMMAND
                        61   11  4D03  5481  BRB T32_S1_X
                                00  4D05  5482 10$: HALT    ; ERROR, unable to Random Repl. Grp 0
                        AB   11  4D06  5483  BRB T32_S1_START  ; Loop on error
                                4D08  5484
                     24    0D   DA  4D08  5485 20$: MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
          08   00000200 8F   DA  4D0B  5486  MTPR # X200,#P0BR  ; Load VA
                     51    3B   DB  4D12  5487  MFPR #TB_DATA,R1  ; Read PTE into R1
                  4D 51    1F   E0  4D15  5488  BBS #TB_VAL,R1,T32_S1_X ; If TB hit, go to next test
       14 0000FE00'EF    1F   E1  4D19  5489  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT
       0000FE44 EF    35'AF   3E  4D21  5490  MOVAW B 30$,APT_ERR_PC ; Save Error PC
          0000FE40'EF    01   88  4D29  5491  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      BF4F   30  4D30  5492  BSBW APT_COMMAND
                        31   11  4D33  5493  BRB T32_S1_X
                                00  4D35  5494 30$: HALT    ; ERROR, unable to Random Repl. Grp. 1
                      FF7A   31  4D36  5495  BRW T32_S1_START  ; Loop on error
                                4D39  5496
                     24    0D   DA  4D39  5497 40$: MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
                     08    00   DA  4D3C  5498  MTPR #0,#P0BR  ; Load VA
                     51    3B   DB  4D3F  5499  MFPR #TB_DATA,R1  ; Read PTE into R1
                  20 51    1F   E0  4D42  5500  BBS #TB_VAL,R1,T32_S1_X ; If TB hit, go to next test
       14 0000FE00'EF    1F   E1  4D46  5501  BBC #31,APT_CTRL_FLAGS,50$ ; Branch if not APT
       0000FE44 EF    62'AF   3E  4D4E  5502  MOVAW B 50$,APT_ERR_PC ; Save Error PC
          0000FE40'EF    01   88  4D56  5503  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      BF22   30  4D5D  5504  BSBW APT_COMMAND
                        04   11  4D60  5505  BRB T32_S1_X
                                00  4D62  5506 50$: HALT    ; ERROR, unable to Random Repl. Grp. 1
                      FF4D   31  4D63  5507  BRW T32_S1_START  ; Loop on error
                                4D66  5508 T32_S1_X:
```

```
                        4D66  5509  .SBTTL MAPPING INITIALIZATION ROUTINE
                        4D66  5510
                        4D66  5511 ;**
                        4D66  5512 ; FUNCTIONAL DESCRIPTION:
                        4D66  5513 ;
                        4D66  5514 ; This routine sets up the Virtual and Physical Address Space
                        4D66  5515 ; necessary for a Mapping test to be performed. Physical Address
                        4D66  5516 ; Space starting at 64 kilobytes will be allocated for the System
                        4D66  5517 ; Page Table(SPT), Program Page Table(P0PT), and Control Page
                        4D66  5518 ; Table(P1PT). All Page Table Entries will be set for Read/Write
                        4D66  5519 ; access. Each Page Table Entry(PTE), will contain Page Frame
                        4D66  5520 ; Numbers(PFN's) from 0 to FF. The following Base and Length
                        4D66  5521 ; Registers will be loaded with information that will allow a
                        4D66  5522 ; System Space Virtual Address, Program Space Virtual Address,
                        4D66  5523 ; and a Control Space Virtual Address to map to the same
                        4D66  5524 ; Physical Address in Main Memory.
                        4D66  5525 ;
                        4D66  5526 ;
                        4D66  5527 ; System Base Register (SBR) = 00010000
                        4D66  5528 ; System Length Register (SLR) =      100
                        4D66  5529 ; P0 Base Register (P0BR) = 80011000
                        4D66  5530 ; P0 Length Register (P0LR) =      100
                        4D66  5531 ; P1 Base Register (P1BR) = 80012000
                        4D66  5532 ; P1 Length Register (P1LR) =        0
                        4D66  5533 ;
                        4D66  5534 ;
                        4D66  5535 ;--
                        4D66  5536 MAP_INIT:
    0C   00010000 8F    DA  4D66  5537  MTPR #SYS_PTBL,#SBR  ; Load PA of SPT into SBR
    0D   00000100 8F    DA  4D6D  5538  MTPR # X100,#SLR  ; Set length of SPT to 100
    08   80011000 8F    DA  4D74  5539  MTPR #P0_PTBL,#P0BR  ; Load PA of P0PT into P0BR
    09   00000100 8F    DA  4D7B  5540  MTPR # X100,#P0LR  ; Set lenght of P0PT to 100
    0A   80012000 8F    DA  4D82  5541  MTPR #P1_PTBL,#P1BR  ; Load PA of P1PT into P1BR
             0B   00    DA  4D89  5542  MTPR #0,#P1LR  ; Set P1LR for no unmapped pages
             55   00    D0  4D8C  5543  MOVL #0,R5  ; Initialize counter
    51   A4000000 8F    D0  4D8F  5544  MOVL # XA4000000,R1  ; Load full access PTE into R1
    52   00010000 8F    D0  4D96  5545  MOVL #SYS_PTBL,R2  ; Initialize R2 with PA of SPT
    53   00011000 8F    D0  4D9D  5546  MOVL # X11000,R3  ; Initialize R3 with PA of P0PT
    54   00012000 8F    D0  4DA4  5547  MOVL # X12000,R4  ; Initialize R4 with PA of P1PT
             82   51    D0  4DAB  5548 70$: MOVL R1,(R2)+  ; Load PTE into SPT
             83   51    D0  4DAE  5549  MOVL R1,(R3)+  ; Load PTE into P0PT
             84   51    D0  4DB1  5550  MOVL R1,(R4)+  ; Load PTE into P1PT
                  51    D6  4DB4  5551  INCL R1  ; Generate next PTE
ED 55   000000FF 8F    F3  4DB6  5552  AOBLEQ # XFF,R5,70$  ; Continue for 256 PTE S
                        4DBE  5553
                0000    31  4DBE  5554  BRW TEST_33  ; Go to next test
```

```
                        4DC1  5555  .SBTTL  TEST 33: TB MAPPING TEST
                        4DC1  5556 ;++
                        4DC1  5557 ; FUNCTIONAL DESCRIPTION:
                        4DC1  5558 ;
                        4DC1  5559 ; This test will verify proper TB operation with Memory Mapping
                        4DC1  5560 ; enabled. Three Virtual Addresses, VA1, VA2, and VA3, will be
                        4DC1  5561 ; referenced which will all map to Physical Address 18000.
                        4DC1  5562 ;
                        4DC1  5563 ;
                        4DC1  5564 ; IMPLICIT INPUTS:
                        4DC1  5565 ;
                        4DC1  5566 ; VA1::  .LONG  X00018000 VA Program Space
                        4DC1  5567 ; VA2::  .LONG  X80018000 VA System Space
                        4DC1  5568 ; VA3::  .LONG  X40018000 VA Control Space
                        4DC1  5569 ;   .LONG  X00018000 VA Program Space
                        4DC1  5570 ;   .LONG  X80018000 VA System Space
                        4DC1  5571 ;   .LONG   X00018000 VA Program Space
                        4DC1  5572 ;   .LONG  X40018000 VA Control Space
                        4DC1  5573 ;
                        4DC1  5574 ;
                        4DC1  5575 ; TB_PAT_D:: .LONG  XAAAAAAAA TB patter· 1
                        4DC1  5576 ;   .LONG  X55555555 TB pattern 2
                        4DC1  5577 ;   .LONG  XF0F0F0F0 TB pattern 3
                        4DC1  5578 ;
                        4DC1  5579 ;
                        4DC1  5580 ;
                        4DC1  5581 ;--
                        4DC1  5582
                        4DC1  5583
                        4DC1  5584  SEQ  $TN   ; Macro Call for Test Sequence Check
                        4DC1       TEST_33::    ; Continued on next line
          0000FE50 EF   21   B0   4DC1         MOVW #33,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF   B6   4DC8         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50 EF   B1   4DCE         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52 EF        4DD4          APT_TEST_COUNT  ; Check for the correct test number
                        20   13   4DD9         BEQL T33_S1   ; Sequence correct, continue test
          0000FE00 EF   1F   E1   4DDB         BBC #31,APT_CTRL_FLAGS,-
                        17        4DE2          T33_HLT   ; Branch if not APT mode
0000FE44 EF   00004DFA EF   9E   4DE3         MOVAB T33_HLT,APT_ERR_PC ; Save the error PC
          0000FE40 EF   01   B0   4DEE         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                     BE8A   30   4DF5         BSBW APT_COMMAND  ; Call APT routine
                        01   11   4DF8         BRB T33_S1   ; Continue
                             4DFA        T33_HLT:
                        00   4DFA         HALT    ; ERROR!!! Program out of sequence
                             4DFB        T33_S1:
           BC06 CF    FD AF   DE   4DFB         MOVAL T33_S1,RET_PC  ; Store return PC
                             4E01
                             4E01  5585
                             4E01  5586 ;
                             4E01  5587 ; Subtest 1 checks that the correct Physical Address is referenced in
                             4E01  5588 ; Main Memory. This is accomplished by initially writing all ones in
                             4E01  5589 ; PA 00018000, and then enabling Memory Mapping. VA1-3 are then read
                             4E01  5590 ; and checked for correct data results.
                             4E01  5591 ;
                             4E01  5592
                   26   0F   DA   4E01  5593  MTPR # XF,#MCESR  ; Clear MCESR
                   39   00   DA   4E04  5594  MTPR #0,#TBIA  ; Flush the TB
```

```
                    24    0A    DA   4E07   5595  MTPR #XA,#TBDR  ; Replace Grp. 0 disable Grp. 1
                    54    01    D0   4E0A   5596  MOVL #1,R4    ; Initialize counter
                50  B537  CF    DE   4E0D   5597  MOVAL VA1,R0    ; Load Virtual Address pointer
          00 B0   FFFFFFFF 8F    D0   4E12   5598  MOVL #XFFFFFFFF,a(R0) ; Load data=all ones into PA 18000
                    38    01    DA   4E1A   5599  MTPR #1,#MME    ; Enable Mapping
          00 B0   FFFFFFFF 8F    D1   4E1D   5600 10$: CMPL #XFFFFFFFF,a(R0) ; Check if mapped to correct PA
                           1F    13   4E25   5601  BEQL 30$    ; If PA correct continue test
          14 0000FE00 EF   1F    E1   4E27   5602  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
          0000FE44'EF    43 AF    3E   4E2F   5603  MOVAW B 20$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88   4E37   5604  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       BE41  30    4E3E   5605  BSBW APT_COMMAND
                          03    11   4E41   5606  BRB 30$
                                00   4E43   5607 20$: HALT     ; Did not reference correct PA=18000
                                     4E44   5608       ; R0 contains pointer to ref. VA
                          D7    11   4E44   5609  BRB 10$    ; Loop on error
                    50    04    C0   4E46   5610 30$: ADDL2 #4,R0    ; Point to next VA
                  D0 54   03    F3   4E49   5611  AOBLEQ #3,R4,10$  ; Repeat for three VA's
                                     4E4D   5612
                                     4E4D   5613 ;
                                     4E4D   5614 ; Subtest 2 checks that by writing Test Pattern 1 into VA1, the same
                                     4E4D   5615 ; pattern will be read when referencing VA2 and VA3. This is repeated
                                     4E4D   5616 ; for writing Test Pattern 2 into VA3, while reading VA1 and VA2.
                                     4E4D   5617 ; Finally Test Pattern 3 is written into VA2, while VA1 and VA3 are
                                     4E4D   5618 ; checked for correct data.
                                     4E4D   5619 ;
                                     4E4D   5620
           BBB4 CF    FD AF    DE   4E4D   5621 T33_S2: MOVAL T33_S2,RET_PC  ; Store Return PC
                    39    00    DA   4E53   5622  MTPR #0,#TBIA  ; Flush the TB
                    54    01    D0   4E56   5623  MOVL #1,R4    ; Initialize counter
                51  B4DF  CF    DE   4E59   5624  MOVAL TB_PAT_D,R1  ; Load addr. of data pattern into R1
                52  B4E6  CF    DE   4E5E   5625  MOVAL VA1,R2    ; Load pointer to VA
                    92    61    D0   4E63   5626 10$: MOVL (R1),a(R2)+  ; Write pattern into VA
                  00 B2   61    D1   4E66   5627 20$: CMPL (R1),a(R2)  ; Read contents of next VA
                           1F    13   4E6A   5628  BEQL 40$    ; If data correct continue test
          14 0000FE00 EF   1F    E1   4E6C   5629  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
          0000FE44 EF    88 AF    3E   4E74   5630  MOVAW B 30$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88   4E7C   5631  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       BDFC  30    4E83   5632  BSBW APT_COMMAND
                          03    11   4E86   5633  BRB 40$
                                00   4E88   5634 30$: HALT     ; ERROR, did not map to correct PA
                                     4E89   5635       ; when reading VA.
                                     4E89   5636       ; R1 contains pointer to exp. data
                                     4E89   5637       ; R2 contains pointer to ref. VA
                          DB    11   4E89   5638  BRB 20$    ; Loop on error
                    52    04    C0   4E8B   5639 40$: ADDL2 #4,R2    ; Generate next VA
                  00 B2   61    D1   4E8E   5640 45$: CMPL (R1),a(R2)  ; Read contents of next VA
                           1F    13   4E92   5641  BEQL 60$    ; If data correct continue test
          14 0000FE00 EF   1F    E1   4E94   5642  BBC #31,APT_CTRL_FLAGS,50$ ; Branch if not APT mode
          0000FE44 EF    B0 AF    3E   4E9C   5643  MOVAW B 50$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88   4EA4   5644  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       BDD4  30    4EAB   5645  BSBW APT_COMMAND
                          03    11   4EAE   5646  BRB 60$
                                00   4EB0   5647 50$. HALT     ; ERROR, did not map to correct PA
                                     4EB1   5648       ; when reading VA.
                                     4EB1   5649       ; R1 contains pointer to exp. data
                                     4EB1   5650       ; R2 contains pointer to ref. VA
                          DB    11   4EB1   5651  BRB 45$    ; Loop on error
```

```
                51    04    C0   4EB3   5652 60$: ADDL2 #4,R1    ; Generate next data pattern
              A9 54    03    F3   4EB6   5653  AOBLEQ #3,R4,10$  ; Continue for VA1 through VA3
                                  4EBA   5654
                                  4EBA   5655 ;
                                  4EBA   5656 ; Subtest 3 is identical to subtest 2 except that force replacement
                                  4EBA   5657 ; is done on TB Group 1, and Group 0 is disabled.
                                  4EBA   5658 ;
                                  4EBA   5659
         BB47 CF    FD AF    DE   4EBA   5660 T33_S3: MOVAL T33_S3,RET_PC  ; Store Return PC
                39    00    DA   4EC0   5661  MTPR #0,#TBIA  ; Flush the TB
                54    01    D0   4EC3   5662  MOVL #1,R4    ; Initialize counter
                24    0D    DA   4EC6   5663  MTPR # XD,#TBDR  ; Replace Grp. 1, disable Grp. 0
             51    B46F CF    DE   4EC9   5664  MOVAL TB_PAT_D,R1  ; Load addr. of data pattern into R1
             52    B476 CF    DE   4ECE   5665  MOVAL VA1,R2    ; Load pointer to VA
                92    61    D0   4ED3   5666 10$: MOVL (R1),a(R2)+  ; Write pattern into VA
               00 B2    61    D1   4ED6   5667 20$: CMPL (R1),a(R2)  ; Read contents of next VA
                      1F    13   4EDA   5668  BEQL 40$    ; If data correct, continue test
     14 0000FE00 EF    1F    E1   4EDC   5669  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
     0000FE44 EF    F8'AF    3E   4EE4   5670  MOVAW B 30$,APT_ERR_PC ; Save Error PC
        0000FE40 EF    01    88   4EEC   5671  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                    BD8C    30   4EF3   5672  BSBW APT_COMMAND
                      03    11   4EF6   5673  BRB 40$
                            00   4EF8   5674 30$: HALT     ; ERROR, did not map to correct PA
                                  4EF9   5675       ; when reading VA.
                                  4EF9   5676       ; R1 contains pointer to exp. data
                                  4EF9   5677       ; R2 contains pointer to ref. VA
                      DB    11   4EF9   5678  BRB 20$    ; Loop on error
                52    04    C0   4EFB   5679 40$: ADDL2 #4,R2    ; Generate next VA
               00 B2    61    D1   4EFE   5680 45$: CMPL (R1),a(R2)  ; Read contents of next VA
                      1F    13   4F02   5681  BEQL 60$    ; If data correct, continue test
     14 0000FE00 EF    1F    E1   4F04   5682  BBC #31,APT_CTRL_FLAGS,50$ ; Branch if not APT mode
     0000FE44 EF    20'AF    3E   4F0C   5683  MOVAW B 50$,APT_ERR_PC ; Save Error PC
        0000FE40'EF    01    88   4F14   5684  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                    BD64    30   4F1B   5685  BSBW APT_COMMAND
                      03    11   4F1E   5686  BRB 60$
                            00   4F20   5687 50$: HALT     ; ERROR, did not map to correct PA
                                  4F21   5688       ; when reading VA.
                                  4F21   5689       ; R1 contains pointer to exp. data
                                  4F21   5690       ; R2 contains pointer to ref. VA
                      DB    11   4F21   5691  BRB 45$    ; Loop on error
                51    04    C0   4F23   5692 60$: ADDL2 #4,R1    ; Generate next data pattern
              A9 54    03    F3   4F26   5693  AOBLEQ #3,R4,10$  ; Continue for writing 3 VA's.
                                  4F2A   5694
                                  4F2A   5695
                                  4F2A   5696 ;
                                  4F2A   5697 ; Subtest 4 is identical to subtest 2 except that random replacement
                                  4F2A   5698 ; is used.
                                  4F2A   5699 ;
                                  4F2A   5700
         BAD7 CF    FD AF    DE   4F2A   5701 T33_S4: MOVAL T33_S4,RET_PC  ; Store Return PC
                39    00    DA   4F30   5702  MTPR #0,#TBIA  ; Flush the TB
                54    01    D0   4F33   5703  MOVL #1,R4    ; Initialize counter
                24    00    DA   4F36   5704  MTPR #0,#TBDR  ; Enable random replacement
             51    B3FF CF    DE   4F39   5705  MOVAL TB_PAT_D,R1  ; Load addr. of data pattern into R1
             52    B406 CF    DE   4F3E   5706  MOVAL VA1,R2    ; Load pointer to VA
                92    61    D0   4F43   5707 10$: MOVL (R1),a(R2)+  ; Write pattern into VA
               00 B2    61    D1   4F46   5708 20$: CMPL (R1),a(R2)  ; Read contents of next VA
```

```
                    1F   13  4F4A  5709  BEQL 40$    ; If data correct, continue test
          14 0000FE00 EF   1F  E1  4F4C  5710  BBC #31,APT_CTRL_FLAGS,30$ ; Branch if not APT mode
          0000FE44 EF   68'AF  3E  4F54  5711  MOVAW B 30$,APT_ERR_PC ; Save Error PC
             0000FE40 EF   01  88  4F5C  5712  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        BD1C  30  4F63  5713  BSBW APT_COMMAND
                          03  11  4F66  5714  BRB 40$
                              00  4F68  5715 30$: HALT     ; ERROR, did not map to correct PA
                                  4F69  5716       ; when reading VA.
                                  4F69  5717       ; R1 contains pointer to exp. data
                                  4F69  5718       ; R2 contains pointer to ref. VA
                          DB  11  4F69  5719  BRB 20$    ; Loop on error
                       52   04  C0  4F6B  5720 40$: ADDL2 #4,R2    ; Generate next VA
                    00 B2   61  D1  4F6E  5721 45$: CMPL (R1),@(R2)  ; Read contents of next VA
                          1F  13  4F72  5722  BEQL 60$    ; If data correct, continue test
          14 0000FE00 EF   1F  E1  4F74  5723  BBC #31,APT_CTRL_FLAGS,50$ ; Branch if not APT mode
          0000FE44'EF   90'AF  3E  4F7C  5724  MOVAW B 50$,APT_ERR_PC ; Save Error PC
             0000FE40 EF   01  88  4F84  5725  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                        BCF4  30  4F8B  5726  BSBW APT_COMMAND
                          03  11  4F8E  5727  BRB 60$
                              00  4F90  5728 50$: HALT     ; ERROR, did not map to correct PA
                                  4F91  5729       ; when reading VA.
                                  4F91  5730       ; R1 contains pointer to exp. data
                                  4F91  5731       ; R2 contains pointer to ref. VA
                          DB  11  4F91  5732  BRB 45$    ; Loop on error
                       51   04  C0  4F93  5733 60$: ADDL2 #4,R1    ; Generate next data pattern
                    A9 54   03  F3  4F96  5734  AOBLEQ #3,R4,10$  ; Continue for writing 3 VA's.
                       38   00  DA  4F9A  5735  MTPR #0,#MME    ; Disable Mapping
```

```
                                4F9D  5736  .SBTTL TEST 34: TB MODIFIED BIT TEST
                                4F9D  5737
                                4F9D  5738 ;++
                                4F9D  5739 ; FUNCTIONAL DESCRIPTION:
                                4F9D  5740 ;
                                4F9D  5741 ; This test verifies that the Modified bit will set on a
                                4F9D  5742 ; valid, access allowed write to memory.
                                4F9D  5743 ;
                                4F9D  5744 ;
                                4F9D  5745 ;--
                                4F9D  5746
                                4F9D  5747
                                4F9D  5748  SEQ  $TN    ; Macro call for Test Sequence Check
                                4F9D        TEST_34::     ; Continued on next line
          0000FE50'EF    22    B0  4F9D         MOVW #34,APT_TEST_NUM ; Load the Test Number
               0000FE52'EF    B6  4FA4         INCW APT_TEST_COUNT  ; Bump up the test count
               0000FE50'EF    B1  4FAA         CMPW APT_TEST_NUM,-  ; Continued on next line
               0000FE52'EF        4FB0          APT_TEST_COUNT  ; Check for the correct test number
                     20    13  4FB5         BEQL T34_S1    ; Sequence correct, continue test
          0000FE00 EF    1F  E1  4FB7         BBC #31,APT_CTRL_FLAGS,-
                     17        4FBE          T34_HLT    ; Branch if not APT mode
     0000FE44'EF    00004FD6'EF  9E  4FBF         MOVAB T34_HLT,APT_ERR_PC ; Save the error PC
          0000FE40'EF    01  B0  4FCA         MOVW #1,APT_ERR_FLAG  ; Set the error flag
                   BCAE    30  4FD1         BSBW APT_COMMAND  ; Call APT routine
                     01    11  4FD4         BRB T34_S1    ; Continue
                           4FD6        T34_HLT:
                           00  4FD6         HALT     ; ERROR!!! Program out of sequence
                           4FD7        T34_S1:
           BA2A CF    FD AF  DE  4FD7         MOVAL T34_S1,RET_PC  ; Store return PC
                           4FDD
                     26    0F  DA  4FDD  5749  MTPR # XF,#MCESR  ; Clear MCESR
                     39    00  DA  4FE0  5750  MTPR #0,#TBIA  ; Flush the TB
                     24    0A  DA  4FE3  5751  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
00010300 EF    A00000C0  8F  D0  4FE6  5752  MOVL # XA00000C0,SPTE ; Write PTE with M bit-clear into SPT
                     38    01  DA  4FF1  5753 10$: MTPR #1,#MME    ; Enable Mapping
                B353 DF    00  D0  4FF4  5754  MOVL #0,aVA2    ; Write into page with M-bit clear
                     38    00  DA  4FF9  5755  MTPR #0,#MME    ; Disable Mapping
          51    00010300'EF  D0  4FFC  5756  MOVL SPTE,R1    ; Read PTE into R1
                  1F 51    1A  E0  5003  5757  BBS #26,R1,30$  ; If M-Bit set, proceed to next test
        14 0000FE00'EF    1F  E1  5007  5758  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
        0000FE44'EF    23'AF  3E  500F  5759  MOVAW B 20$,APT_ERR_PC ; Save Error PC
           0000FE40 EF    01  88  5017  5760  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                    BC61    30  501E  5761  BSBW APT_COMMAND
                     03    11  5021  5762  BRB 30$
                           00  5023  5763 20$: HALT     ; ERROR, Modified Bit did not set.
                               5024  5764       ; R1 bit<26> contains M-Bit of
                               5024  5765       ; referenced PTE
                     CB    11  5024  5766  BRB 10$    ; Loop on error
                               5026  5767 30$:      ; Go to next test
```

```
                                5026  5768  .SBTTL TEST 35: TB CROSS PAGE REFERENCE TEST
                                5026  5769 ;++
                                5026  5770 ; FUNCTIONAL DESCRIPTION:
                                5026  5771 ;
                                5026  5772 ; This test makes memory references across two pages
                                5026  5773 ; designated as VPN1 (VA=00018000) and VPN2 (VA=00018200).
                                5026  5774 ; Longword references to the last two bytes in VPN1 will
                                5026  5775 ; cause the first two bytes in VPN2 to be referenced. Using
                                5026  5776 ; this technique, each page will be set up with various
                                5026  5777 ; access rights and the appropriate response will be
                                5026  5778 ; be checked.
                                5026  5779 ;
                                5026  5780 ; The PTE for VPN1 is at PA=00011300
                                5026  5781 ; The PTE for VPN2 is at PA=00011304
                                5026  5782 ;
                                5026  5783 ; Prior to making any cross page reference, an Access Control
                                5026  5784 ; Violation Flag(known as ACC_VIO_FLAG) will be set. The
                                5026  5785 ; reference will then be attempted. If an Access Control
                                5026  5786 ; Violation(ACV) is expected, the Service Routine will
                                5026  5787 ; clear the ACC_VIO_FLAG and return from the exception.
                                5026  5788 ; The test routine will check the flag, if clear, it
                                5026  5789 ; indicates that the expected Access Control Violation
                                5026  5790 ; did in fact, occur. If the flag was still set, the
                                5026  5791 ; exception never occured and this would cause an
                                5026  5792 ; error halt.
                                5026  5793 ;
                                5026  5794 ;
                                5026  5795 ;
                                5026  5796 ; IMPLICIT INPUTS:
                                5026  5797 ;
                                5026  5798 ; SPTE  ==  X10300  Address for PTE in SPT
                                5026  5799 ; SYS_PTBL ==  X10000  Physical loc. of SPT
                                5026  5800 ; P0_PTBL  ==  X80011000 Virtual loc. of P0PT
                                5026  5801 ; P1_PTBL  ==  X80012000 Virtual loc. of P1PT
                                5026  5802 ; P0PTE_VPN1 ==  X11300  Physical loc. of PTE
                                5026  5803 ; P0PTE_VPN2 ==  X11304  Physical loc. of PTE
                                5026  5804 ; END_VPN1 ==  X181FE  Byte 511 of VPN1
                                5026  5805 ;
                                5026  5806 ;
                                5026  5807 ;--
                                5026  5808
                                5026  5809
                                5026  5810  SEQ  $TN    ; Macro Call for Test Sequence Check
                                5026       TEST_35::     ; Continued on next line
        0000FE50 EF    23   B0  5026        MOVW #35,APT_TEST_NUM ; Load the Test Number
             0000FE52 EF    B6  502D        INCW APT_TEST_COUNT  ; Bump up the test count
             0000FE50'EF    B1  5033        CMPW APT_TEST_NUM,-  ; Continued on next line
             0000FE52'EF        5039         APT_TEST_COUNT  ; Check for the correct test number
                        20  13  503E        BEQL T35_S1    ; Sequence correct, continue test
        0000FE00'EF    1F  E1  5040        BBC #31,APT_CTRL_FLAGS,-
                        17      5047         T35_HLT    ; Branch if not APT mode
0000FE44 EF    0000505F EF  9E  5048        MOVAB T35_HLT,APT_ERR_PC ; Save the error PC
        0000FE40 EF    01   B0  5053        MOVW #1,APT_ERR_FLAG  ; Set the error flag
                     BC25   30  505A        BSBW APT_COMMAND  ; Call APT routine
                       01   11  505D        BRB T35_S1    ; Continue
                            505F       T35_HLT:
                        00  505F        HALT    ; ERROR!!! Program out of sequence
```

```
                                5060        T35_S1:
             B9A1 CF    FD AF    DE  5060         MOVAL T35_S1,RET_PC  ; Store return PC
                                5066
                                5066  5811 ;
                                5066  5812 ; Subtest 1 does a write and read of a longword of A's across
                                5066  5813 ; two pages set up with read/write access. No Access Control
                                5066  5814 ; Violation should occur.
                                5066  5815 ;
                                5066  5816
                    26    0F    DA  5066  5817  MTPR # XF,#MCESR  ; Clear MCESR
                    39    00    DA  5069  5818  MTPR #0,#TBIA  ; Flush the TB
                    24    0A    DA  506C  5819  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
               B992 CF    01    88  506F  5820  BISB2 #1,ACC_VIO_FLAG  ; Set Access Violation Flag
    00011300'EF   A40000C0 8F    D0  5074  5821  MOVL # XA40000C0,P0PTE_VPN1 ; Load PTE for VPN1 into P0PT
    00011304'EF   A40000C1 8F    D0  507F  5822  MOVL # XA40000C1,P0PTE_VPN2 ; Load PTE for VPN2 into P0PT
               B977 CF    01    88  508A  5823 10$: BISB2 #1,ACC_VIO_FLAG  ; Set Access Violation Flag
                    38    01    DA  508F  5824  MTPR #1,#MME  ; Enable Mapping
    000181FE'EF   AAAAAAAA 8F    D0  5092  5825  MOVL # XAAAAAAAA,END_VPN1 ; Write Longword of A's at byte 511
                    38    00    DA  509D  5826  MTPR #0,#MME  ; Disable Mapping
             1F B961 CF    00    E0  50A0  5827  BBS #0,ACC_VIO_FLAG,30$ ; If no ACV, continue test
          14 0000FE00'EF    1F    E1  50A6  5828  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
          0000FE44 EF    C2'AF    3E  50AE  5829  MOVAW B 20$,APT_ERR_PC ; Save Error PC
             0000FE40'EF    01    88  50B6  5830  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       BBC2    30  50BD  5831  BSBW APT_COMMAND
                         03    11  50C0  5832  BRB 30$
                               00  50C2  5833 20$: HALT  ; ERROR unexpected ACV occured
                         C5    11  50C3  5834  BRB 10$  ; Loop on error
               B93C CF    01    88  50C5  5835 30$: BISB2 #1,ACC_VIO_FLAG  ; Set Access Violation Flag
                    38    01    DA  50CA  5836  MTPR #1,#MME  ; Enable Mapping
            51    000181FE'EF    D0  50CD  5837  MOVL END_VPN1,R1  ; Read longword of A's at byte 511
                    38    00    DA  50D4  5838  MTPR #0,#MME  ; Disable Mapping
             27 B92A CF    00    E0  50D7  5839  BBS #0,ACC_VIO_FLAG,50$ ; If no ACV, continue test
    0000FE40'EF    000050FC'EF    3E  50DD  5840  MOVAW 40$,APT_ERR_FLAG ; Save Error PC
          0C 0000FE00'EF    1F    E1  50E8  5841  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
             0000FE40'EF    01    88  50F0  5842  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       BB88    30  50F7  5843  BSBW APT_COMMAND
                         08    11  50FA  5844  BRB 50$
                               0C  50FC  5845 40$: HALT  ; ERROR, unexpected ACV occured
               B904 CF    01    88  50FD  5846  BISB2 #1,ACC_VIO_FLAG  ; Reset ACV flag
                         C1    11  5102  5847  BRB 30$  ; Loop on error
            51    AAAAAAAA 8F    D1  5104  5848 50$: CMPL # XAAAAAAAA,R1  ; Check data read at VPN1
                         20    13  510B  5849  BEQL T35_S2  ; If data correct, go to next subtest
          14 0000FE00 EF    1F    E1  510D  5850  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
          0000FE44 EF    29'AF    3E  5115  5851  MOVAW B 60$,APT_ERR_PC ; Save Error PC
             0000FE40 EF    01    88  511D  5852  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                       BB5B    30  5124  5853  BSBW APT_COMMAND
                         04    11  5127  5854  BRB 70$
                               00  5129  5855 60$: HALT  ; ERROR, data compare incorrect
                                   512A  5856      ; expected data=AAAAAAAA
                                   512A  5857      ; R1 contains data read.
                       FF5D    31  512A  5858  BRW 10$  ; Loop on error
                                   512D  5859 70$:
                                   512D  5860
                                   512D  5861
                                   512D  5862
                                   512D  5863 ;
                                   512D  5864 ; Subtest 2 attempts to write a longword of 5 s across two
```

```
                                512D  5865 ; pages.  VP1 is set up for read/write access, and VP2 is read
                                512D  5866 ; only. An Access Control Violation should occur and the write
                                512D  5867 ; should not be allowed. A read is then attempted across both
                                512D  5868 ; pages which should be allowed, and the data results should
                                512D  5869 ; still indicate a pattern of A's which were written from the
                                512D  5870 ; previous subtest.
                                512D  5871 ;
                                512D  5872
                                512D  5873
            B8D4 CF    FD AF    DE  512D  5874 T35_S2: MOVAL T35_S2,RET_PC  ; Store Return PC
                   26    0F    DA  5133  5875  MTPR #^XF,#MCESR  ; Clear MCESR
                   39    00    DA  5136  5876  MTPR #0,#TBIA  ; Flush the TB
                   24    0A    DA  5139  5877  MTPR #^XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
     0C011300 EF    FC0000C0 8F    D0  513C  5878  MOVL #^XFC0000C0,@PTE_VPN1 ; Make VPN1 read only
                B8BA CF    01    88  5147  5879  BISB2 #1,ACC_VIO_FLAG  ; Set Access Violation Flag
                   38    01    DA  514C  5880 10$: MTPR #1,#MME  ; Enable Mapping
        B8B3 CF    00005163'EF    DE  514F  5881  MOVAL 15$,CONT_PC  ; Store continue Addr. for exp. ACV
     000181FE'EF    55555555 8F    D0  5158  5882  MOVL #^X55555555,END_VPN1 ; Write a longword of 5's at byte 511
                   38    00    DA  5163  5883 15$: MTPR #0,#MME  ; Disable Mapping
              1F B89B CF    00    E1  5166  5884  BBC #0,ACC_VIO_FLAG,30$ ; If ACV occured, continue test
           14 0000FE00'EF    1F    E1  516C  5885  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
           0000FE44'EF    88'AF    3E  5174  5886  MOVAW B 20$,APT_ERR_PC ; Save Error PC
              0000FE40'EF    01    88  517C  5887  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      BAFC    30  5183  5888  BSBW APT_COMMAND
                         03    11  5186  5889  BRB 30$
                         00    5188  5890 20$: HALT  ; ERROR, did not get expected ACV
                         C1    11  5189  5891  BRB 10$  ; Loop on error
              B876 CF    01    88  518B  5892 30$: BISB2 #1,ACC_VIO_FLAG  ; Set Access Violation Flag
                   38    01    DA  5190  5893  MTPR #1,#MME  ; Enable Mapping
        51    000181FE'EF    D0  5193  5894  MOVL END_VPN1,R1  ; Read a longword at byte 511
                   38    00    DA  519A  5895  MTPR #0,#MME  ; Disable Mapping
         1F B864 CF    00    E0  519D  5896  BBS #0,ACC_VIO_FLAG,50$ ; If no ACV occured, continue test
     14 0000FE00'EF    1F    E1  51A3  5897  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
     0000FE44'EF    BF'AF    3E  51AB  5898  MOVAW B 40$,APT_ERR_PC ; Save Error PC
        0000FE40'EF    01    88  51B3  5899  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      BAC5    30  51BA  5900  BSBW APT_COMMAND
                         03    11  51BD  5901  BRB 50$
                         00    51BF  5902 40$: HALT  ; ERROR, unexpected ACV occured
                         C9    11  51C0  5903  BRB 30$  ; Loop on error
        51    AAAAAAAA 8F    D1  51C2  5904 50$: CMPL #^XAAAAAAAA,R1  ; Check data read at VPN1
                         20    13  51C9  5905  BEQL T35_S3  ; If data correct go to next subtest
     14 0000FE00 EF    1F    E1  51CB  5906  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
     0000FE44 EF    E7 AF    3E  51D3  5907  MOVAW B 60$,APT_ERR_PC ; Save Error PC
        0000FE40 EF    01    88  51DB  5908  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                      BA9D    30  51E2  5909  BSBW APT_COMMAND
                         04    11  51E5  5910  BRB 70$
                         00    51E7  5911 60$: HALT  ; ERROR, data compare incorrect
                               51E8  5912  ; expected data=AAAAAAAA
                               51E8  5913  ; R1 contains data read
                      FF61    31  51E8  5914  BRW 10$  ; Loop on error
                               51EB  5915 70$:
                               51EB  5916
                               51EB  5917 ;
                               51EB  5918 ; Subtest 3 attempts to write a longword of 5 s across two pages.
                               51EB  5919 ; VP1 is set up for read only access, and VP2 is read/write access.
                               51EB  5920 ; An Access Control Violation should occur and the data should not
                               51EB  5921 ; be written. A read is then attempted to verify the previons data
```

```
                                    51EB  5922 ; pattern of A's.
                                    51EB  5923 ;
                                    51EB  5924
                                    51EB  5925
          B816 CF    FD AF    DE  51EB  5926 T35_S3: MOVAL T35_S3,RET_PC  ; Store Return PC
                  26    0F    DA  51F1  5927  MTPR # XF,#MCESR  ; Clear MCESR
                  39    00    DA  51F4  5928  MTPR #0,#TBIA  ; Flush the TB
                  24    0A    DA  51F7  5929  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp. 1
    00011300 EF    FC0000C0 8F    D0  51FA  5930  MOVL # XFC0000C0,P0PTE_VPN1 ; Make VPN1 read only
    00011304 EF    A40000C1 8F    D0  5205  5931  MOVL # XA40000C1,P0PTE_VPN2 ; Make VPN2 read/write
              B7F1 CF    01    88  5210  5932  BISB2 #1,ACC_VIO_FLAG  ; Set ACV Flag
                  38    01    DA  5215  5933 10$: MTPR #1,#MME  ; Enable Mapping
        B7EA CF    0000522C'EF    DE  5218  5934  MOVAL 15$,CONT_PC  ; Store continue Addr. for exp. ACV
    000181FE EF    55555555 8F    D0  5221  5935  MOVL # X55555555,END_VPN1 ; Write a longword of 5's at byte 511
                  38    00    DA  522C  5936 15$: MTPR #0,#MME  ; Disable Mapping
           1F B7D2 CF    00    E1  522F  5937  BBC #0,ACC_VIO_FLAG,30$ ; If ACV occurs, continue test
        14 0000FE00'EF    1F    E1  5235  5938  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
        0000FE44 EF    51'AF    3E  523D  5939  MOVAW B 20$,APT_ERR_PC ; Save Error PC
           0000FE40'EF    01    88  5245  5940  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                     BA33    30  524C  5941  BSBW APT_COMMAND
                       03    11  524F  5942  BRB 30$
                             00  5251  5943 20$: HALT  ; ERROR, did not get ACV
                       C1    11  5252  5944  BRB 10$  ; Loop on error
              B7AD CF    01    88  5254  5945 30$: BISB2 #1,ACC_VIO_FLAG  ; Set ACV Flag
                  38    01    DA  5259  5946  MTPR #1,#MME  ; Enable Mapping
           51    000181FE EF    D0  525C  5947  MOVL END_VPN1,R1  ; Read longword at byte 511 of VPN1
                  38    00    DA  5263  5948  MTPR #0,#MME  ; Disable Mapping
           1F B79B CF    00    E0  5266  5949  BBS #0,ACC_VIO_FLAG,50$ ; If no ACV occurs, continue test
        14 0000FE00'EF    1F    E1  526C  5950  BBC #31,APT_CTRL_FLAGS,40$ ; Branch if not APT mode
        0000FE44 EF    88'AF    3E  5274  5951  MOVAW B 40$,APT_ERR_PC ; Save Error PC
           0000FE40'EF    01    88  527C  5952  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                     B9FC    30  5283  5953  BSBW APT_COMMAND
                       03    11  5286  5954  BRB 50$
                             00  5288  5955 40$: HALT  ; ERROR, unexpected ACV occured
                       C9    11  5289  5956  BRB 30$  ; Loop on error
           51    AAAAAAAA 8F    D1  528B  5957 50$: CMPL # XAAAAAAAA,R1  ; Check data read starting at byte 511
                       20    13  5292  5958  BEQL T35_S4  ; If data correct, go to next subtest
        14 0000FE00'EF    1F    E1  5294  5959  BBC #31,APT_CTRL_FLAGS,60$ ; Branch if not APT mode
        0000FE44 EF    B0 AF    3E  529C  5960  MOVAW B 60$,APT_ERR_PC ; Save Error PC
           0000FE40'EF    01    88  52A4  5961  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                     B9D4    30  52AB  5962  BSBW APT_COMMAND
                       04    11  52AE  5963  BRB 70$
                             00  52B0  5964 60$: HALT  ; ERROR, data incorrect
                                 52B1  5965  ; expected data=AAAAAAAA
                                 52B1  5966  ; R1 contains data read
                     FF61    31  52B1  5967  BRW 10$  ; Loop on error
                                 52B4  5968 70$:
                                 52B4  5969
                                 52B4  5970 ;
                                 52B4  5971 ; Subtest 4 attempts to read across two pages. VP1 is set up
                                 52B4  5972 ; with read access only, and VP2 is set up for no access.
                                 52B4  5973 ; An Access Control Violation should occur.
                                 52B4  5974 ;
                                 52B4  5975
                                 52B4  5976
           B74D CF    FD AF    DE  52B4  5977 T35_S4: MOVAL T35_S4,RET_PC  ; Store Return PC
                  26    0F    DA  52BA  5978  MTPR # XF,#MCESR  ; Clear MCESR
```

```
                    39    00    DA  52BD  5979  MTPR #0,#TBIA  ; Flush the TB
                    24    0A    DA  52C0  5980  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Gp. 1
     00011300 EF    FC0000C0 8F   D0  52C3  5981  MOVL # XFC0000C0,P0PTE_VPN1 ; Make VP1 read only
     00011304 EF    840000C1 8F   D0  52CE  5982  MOVL # X840000C1,P0PTE_VPN2 ; Make VP2 no access
                  B728 CF    01   88  52D9  5983  BISB2 #1,ACC_VIO_FLAG  ; Set ACV Flag
                    38    01    DA  52DE  5984 10$: MTPR #1,#MME   ; Enable Mapping
          B721 CF    000052F1'EF   DE  52E1  5985  MOVAL 15$,CONT_PC  ; Store continue Addr. for exp. ACV
               51    000181FE'EF   D0  52EA  5986  MOVL END_VPN1,R1  ; Read longword starting at byte 511
                    38    00    DA  52F1  5987 15$: MTPR #0,#MME   ; Disable Mapping
                1F B70D CF    00   E1  52F4  5988  BBC #0,ACC_VIO_FLAG,T35_S5 ; If ACV occurs, go to next subtest
             14 0000FE00'EF    1F   E1  52FA  5989  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
             0000FE44'EF    16 AF   3E  5302  5990  MOVAW B 20$,APT_ERR_PC ; Save Error PC
                0000FE40'EF    01   88  530A  5991  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         B96E  30   5311  5992  BSBW APT_COMMAND
                           03  11   5314  5993  BRB 50$
                               00   5316  5994 20$: HALT    ; ERROR, ACV did not occur
                           C5  11   5317  5995  BRB 10$   ; Loop on error
                               5319  5996 50$:
                               5319  5997 ;
                               5319  5998 ; Subtest 5 is set up to read across two pages. VP1 is set up
                               5319  5999 ; with no access, and VP2 is set up for read only access. An
                               5319  6000 ; Access Control Violation should occur.
                               5319  6001 ;
                               5319  6002
                               5319  6003
                B6E8 CF    FD AF   DE  5319  6004 T35_S5: MOVAL T35_S5,RET_PC  ; Store Return PC
                         26    0F   DA  531F  6005  MTPR # XF,#MCESR  ; Clear MCESR
                         39    00   DA  5322  6006  MTPR #0,#TBIA  ; Flush the TB
                         24    0A   DA  5325  6007  MTPR # XA,#TBDR  ; Replace Grp. 0, disable Grp.1
     00011300 EF    840000C0 8F   D0  5328  6008  MOVL # X840000C0,P0PTE_VPN1 ; Make VP1 no access
     00011304'EF    FC0000C1 8F   D0  5333  6009  MOVL # XFC0000C1,P0PTE_VPN2 ; Make VP2 read only
                  B6C3 CF    01   88  533E  6010  BISB2 #1,ACC_VIO_FLAG  ; Set ACV Flag
                    38    01    DA  5343  6011 10$: MTPR #1,#MME   ; Enable Mapping
          B6BC CF    00005356'EF   DE  5346  6012  MOVAL 15$,CONT_PC  ; Store continue Addr. for exp. ACV.
               51    000181FE'EF   D0  534F  6013  MOVL END_VPN1,R1  ; Read longword at byte 511 of VP1
                    38    00    DA  5356  6014 15$: MTPR #0,#MME   ; Disable Mapping
                1F B6A8 CF    00   E1  5359  6015  BBC #0,ACC_VIO_FLAG,30$ ; If ACV occurs, print end of pass
             14 0000FE00'EF    1F   E1  535F  6016  BBC #31,APT_CTRL_FLAGS,20$ ; Branch if not APT mode
             0000FE44 EF    7B'AF   3E  5367  6017  MOVAW B 20$,APT_ERR_PC ; Save Error PC
                0000FE40'EF    01   88  536F  6018  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                         B909  30   5376  6019  BSBW APT_COMMAND
                           03  11   5379  6020  BRB 30$
                               00   537B  6021 20$: HALT    ; ERROR, ACV did not occur
                           C5  11   537C  6022  BRB 10$   ; Loop on error
                         25    01   DA  537E  6023 30$: MTPR #1,#CADR  ; Disable Cache for program repeat
                               5381  6024 ; MTPR #3,#TBDR  ; D3.0 Disable TB for program repeat
                               5381  6025
                            B8C8  31   5381  6026  BRW END_PASS  ; Print end of pass msg. and
                               5384  6027       ; repeat test program
                               5384  6028  .SBTTL  INITIALIZATION ROUTINE
                               5384  6029 ;++
                               5384  6030 ; FUNCTIONAL DESCRIPTION:
                               5384  6031 ;
                               5384  6032 ; This routine first loads PCS, then sets up the SCB to point to trap
                               5384  6033 ; catchers, and exception and interrupt routines. The stack pointer is
                               5384  6034 ; also initialized.
                               5384  6035 ; The Machine Check, Access Control Violation, and Write NXM
```

```
                     5384  6036 ; service routines are also contained here.
                     5384  6037 ;
                     5384  6038 ; PCS_INIT - Set-up initial conditions for execution of loader
                     5384  6039 ;
                     5384  6040 ; FUNCTIONAL DESCRIPTION:
                     5384  6041 ;
                     5384  6042 ; This routine will initialize the test conditions.  It sets up the
                     5384  6043 ; necessary stack pointers, SCB entries, and IPR's.
                     5384  6044 ;
                     5384  6045 ; CALLING SEQUENCE:
                     5384  6046 ;
                     5384  6047 ; NONE
                     5384  6048 ;
                     5384  6049 ; INPUT PARAMETERS:
                     5384  6050 ;
                     5384  6051 ; NONE
                     5384  6052 ;
                     5384  6053 ; OUTPUT PARAMETERS:
                     5384  6054 ;
                     5384  6055 ; NONE
                     5384  6056 ;
                     5384  6057 ; COMPLETION CODES:
                     5384  6058 ;
                     5384  6059 ; NONE
                     5384  6060 ;
                     5384  6061 ; SIDE EFFECTS:
                     5384  6062 ;
                     5384  6063 ; NONE
                     5384  6064 ;
                     5384  6065 ;--
                     5384  6066 PCS_INIT::
 52   00005448 9F    DE   5384  6067  MOVAL a#PCS_MCHK_HANDL,R2 ; Load R2 with address of mchk handler
          52   01    C8   538B  6068  BISL2 #BIT0,R2  ; Init the Machine Vector to specify
                     538E  6069       ; using the Interrupt stack.
AF03 CF    B072 CF    D0   538E  6070  MOVL SCB04,PCS_INIT_SAVE ; Save the MCV
     B06A CF    52    D0   5395  6071  MOVL R2,SCB04  ; Init the MCV
```

```
                        539A  6072  .SBTTL PCS_PRESENT - Check for the presense of a PCS (L0008) module
                        539A  6073 ;++
                        539A  6074 ; FUNCTIONAL DESCRIPTION:
                        539A  6075 ;
                        539A  6076 ; This routine will check for the presence of a PCS (L0008) module.  It
                        539A  6077 ; does this by first reading then writing the base address for PCS.  If
                        539A  6078 ; PCS in not present then a machine check will take place (timeout).
                        539A  6079 ; If the PCS is present then the data read back will be checked to see
                        539A  6080 ; that it matches what was written, also, Bit <0> of the PCS status flags
                        539A  6081 ; will be set to signify that a PCS was found.
                        539A  6082 ;
                        539A  6083 ; CALLING SEQUENCE:
                        539A  6084 ;
                        539A  6085 ; NONE
                        539A  6086 ;
                        539A  6087 ; INPUT PARAMETERS:
                        539A  6088 ;
                        539A  6089 ; NONE
                        539A  6090 ;
                        539A  6091 ; OUTPUT PARAMETERS:
                        539A  6092 ;
                        539A  6093 ; PCS_STATUS_FLAGS - Bit 0 will be set if there is a PCS present on the
                        539A  6094 ;      system, clear otherwise.
                        539A  6095 ;
                        539A  6096 ; COMPLETION CODES:
                        539A  6097 ;
                        539A  6098 ; NONE
                        539A  6099 ;
                        539A  6100 ; SIDE EFFECTS:
                        539A  6101 ;
                        539A  6102 ; NONE
                        539A  6103 ;
                        539A  6104 ;--
                        539A  6105
                        539A  6106 PCS_PRESENT::
          52    3E    DB  539A  6107  MFPR #^X3E,R2  ; Get CPU ID Register contents
     52   00FF  8F    AA  539D  6108  BICW2 #^X00FF,R2  ; Clear all but bits 15-8
     5F00 8F    52    B1  53A2  6109  CMPW R2,#^X5F00  ; See if this is a PCS Module
                 0A   18  53A7  6110  BGEQ 1$   ; Branch if so
B054 CF    AEEB  CF    D0  53A9  6111  MOVL PCS_INIT_SAVE,SCB04 ; Restore the MCV
              B6FF   31  53B0  6112  BRW START   ; Exit if not
      00F0C000 9F    D4  53B3  6113 1$: CLRL @#PCS_SELECT_REG ; Try to enable the writing of PCS/WCS
 00000223 9F    01    88  53B9  6114  BISB2 #BIT0,@#PCS_STATUS_FLAGS; Set status bit specifying PCS present
```

```
                        53C0  6115  .SBTTL PCS_BITLOAD - Load the patch bits
                        53C0  6116 ;++
                        53C0  6117 ; FUNCTIONAL DESCRIPTION:
                        53C0  6118 ;
                        53C0  6119 ; This routine will load all 8K of patch bits.  If does a bit test of
                        53C0  6120 ; each patch bit and then writes a 1 or 0 depending upon the outcome of
                        53C0  6121 ; the bit test.
                        53C0  6122 ;
                        53C0  6123 ; CALLING SEQUENCE:
                        53C0  6124 ;
                        53C0  6125 ; NONE
                        53C0  6126 ;
                        53C0  6127 ; INPUT PARAMETERS:
                        53C0  6128 ;
                        53C0  6129 ; NONE
                        53C0  6130 ;
                        53C0  6131 ; OUTPUT PARAMETERS:
                        53C0  6132 ;
                        53C0  6133 ; PCS_STATUS_FLAGS - Bit 1 will be set when all of the patch bits have
                        53C0  6134 ;      been loaded, otherwise 0
                        53C0  6135 ;
                        53C0  6136 ; COMPLETION CODES:
                        53C0  6137 ;
                        53C0  6138 ; NONE
                        53C0  6139 ;
                        53C0  6140 ; SIDE EFFECTS:
                        53C0  6141 ;
                        53C0  6142 ; NONE
                        53C0  6143 ;
                        53C0  6144 ;--
                        53C0  6145
                        53C0  6146 PCS_BITLOAD::
   00F0C000 9F    01    D0  53C0  6147  MOVL #X1,a#PCS_SELECT_REG ; Enable the writing of Patch Bits
                  52    D4  53C7  6148  CLRL R2    ; Init Patch Bit counter
   55    000054A5 EF    DE  53C9  6149  MOVAL PCS_PTH_BIT_ARRAY,R5 ; Load R5 with address of patch bits
   53    00F00000 8F    D0  53D0  6150  MOVL #PCS_PTH_BIT_ADDR,R3 ; Load R3 with patch bit base address
            54    01    D0  53D7  6151  MOVL #X1,R4    ; Init longword mask
  56    65    01    52    EF  53DA  6152 30$: EXTZV R2,#1,(R5),R6  ; Extract the patch bit
            83    56    D0  53DF  6153  MOVL R6,(R3)+  ; Write the patch bit
F0 52    00002000 8F    F2  53E2  6154  AOBLSS #X2000,R2,30$  ; write patch bits until 8K
   00000223'9F    02    88  53EA  6155 100$: BISB2 #BIT1,a#PCS_STATUS_FLAGS; Set bit specifying Patch bits loaded
```

```
                                53F1  6156  .SBTTL PCS_MICLOAD - Load the microcode patches
                                53F1  6157 ;++
                                53F1  6158 ; FUNCTIONAL DESCRIPTION:
                                53F1  6159 ;
                                53F1  6160 ; This routine will load all 1K of microwords into the PCS.  It takes a
                                53F1  6161 ; longword of data and strips the lower 20 bits and writes that to the
                                53F1  6162 ; patch memory.  The first longword is not wrote until the last thing so
                                53F1  6163 ; that bit 20 (PCS enable bit) can be set in that data and then wrote to
                                53F1  6164 ; the patch memory.
                                53F1  6165 ;
                                53F1  6166 ; CALLING SEQUENCE:
                                53F1  6167 ;
                                53F1  6168 ; NONE
                                53F1  6169 ;
                                53F1  6170 ; INPUT PARAMETERS:
                                53F1  6171 ;
                                53F1  6172 ; NONE
                                53F1  6173 ;
                                53F1  6174 ; OUTPUT PARAMETERS:
                                53F1  6175 ;
                                53F1  6176 ; PCS_STATUS_FLAGS - Bit 2 is set when all of the patch microwords are
                                53F1  6177 ;       loaded, otherwise 0.
                                53F1  6178 ;
                                53F1  6179 ; COMPLETION CODES:
                                53F1  6180 ;
                                53F1  6181 ; NONE
                                53F1  6182 ;
                                53F1  6183 ; SIDE EFFECTS:
                                53F1  6184 ;
                                53F1  6185 ; NONE
                                53F1  6186 ;
                                53F1  6187 ;--
                                53F1  6188
                                53F1  6189 PCS_MICLOAD::
      00F0C000 9F    D4         53F1  6190  CLRL @#PCS_SELECT_REG ; Enable the writing of PCS
               52    D4         53F7  6191  CLRL R2    ; Init PCS location counter
 55   000058A5'9F    DE         53F9  6192  MOVAL @#PCS_PTH_MIC_ARRAY,R5 ; Load R5 with PCS array
 53   00F08000 8F    D0         5400  6193  MOVL #PCS_PTH_MIC_ADDR,R3 ; Load R3 with PCS base address
56   65   14    00    EF         5407  6194  EXTZV #0,#20,(R5),R6  ; extract the first 20 bits
           58    14    D0         540C  6195  MOVL #20,R8    ; point position to next 20 bits
           53    04    C0         540F  6196  ADDL2 # X04,R3  ; Point PCS address past first lword
                 52    D6         5412  6197  INCL R2    ; Increment the PCS location counter
83   65   14    58    EF         5414  6198 200$: EXTZV R8,#20,(R5),(R3)+ ; extract 20 bits and write to PCS
           58    14    C0         5419  6199  ADDL2 #20,R8    ; point position to next 20 bits
                 52    D6         541C  6200  INCL R2    ; update pcs location counter
 00001000 8F    52    D1         541E  6201  CMPL R2,-
                                   5425  6202    #PCS_PTH_MIC_SIZE_LWORDS; Check to see if done writing
                 ED    12         5425  6203  BNEQU 200$    ; Branch if NOT done
 56   00100000 8F    C8         5427  6204 210$: BISL2 #BIT20,R6  ; Set the PCS enable bit in first lword
 00F08000 9F    56    D0         542E  6205  MOVL R6,@#PCS_PTH_MIC_ADDR ; Write the first lword to enable PCS
 00000223 9F    04    88         5435  6206  BISB2 #BIT2,@#PCS_STATUS_FLAGS; Set bit specifying Patch bits loaded
AFC1 CF    AE58 CF    B0         543C  6207  MOVW PCS_INIT_SAVE,SCB04 ; Restore MCV
               B66C    31         5443  6208  BRW START    ; All done, branch to rest of init
```

```
5446  6209  .SBTTL PCS_MCHK_HANDL - Machine handler routine.
5446  6210 ;++
5446  6211 ; FUNCTIONAL DESCRIPTION:
5446  6212 ;
5446  6213 ; This routine will handle the machine check.  It will check to see what
5446  6214 ; part of the program caused the machine check and set-up a message
5446  6215 ; address for the print routine to identify the machine check.
5446  6216 ;
5446  6217 ; CALLING SEQUENCE:
5446  6218 ;
5446  6219 ; NONE
5446  6220 ;
5446  6221 ; INPUT PARAMETERS:
5446  6222 ;
5446  6223 ; PCS_STATUS_FLAGS - Used to determine where in the program the mach'ne
5446  6224 ;       check took place.
5446  6225 ;
5446  6226 ; OUTPUT PARAMETERS:
5446  6227 ;
5446  6228 ; NONE
5446  6229 ;
5446  6230 ; COMPLETION CODES:
5446  6231 ;
5446  6232 ; NONE
5446  6233 ;
5446  6234 ; SIDE EFFECTS:
5446  6235 ;
5446  6236 ; NONE
5446  6237 ;
5446  6238 ;--
```

```
                                5446  6239  .ALIGN LONG
                                5448  6240 PCS_MCHK_HANDL::
      12 00000223'9F    00   E0  5448  6241  BBS #0,a#PCS_STATUS_FLAGS,300$
                                5450  6242        ; Check to see if PCS is present
      06 0000FE00'EF    1F   E0  5450  6243  BBS #31,APT_CTRL_FLAGS,10$ ; Go call hard error if APT
                       B8FB  30  5458  6244  BSBW TYPE_NO_PCS  ; Go type message to user if no PCS
                       B600  31  545B  6245  BRW INIT   ; Branch to start test - no PCS module
                       0023  30  545E  6246 10$: BSBW ERROR_CALL  ; Go call error to APT
                             00  5461  6247  HALT    ; PCS is not present
      06 00000223'9F    01   E0  5462  6248 300$: BBS #1,a#PCS_STATUS_FLAGS,310$
                                546A  6249        ; Check to see if Patch bits loaded
                       0017  30  546A  6250  BSBW ERROR_CALL  ; Go call the error
                             00  546D  6251 25$: HALT    ; Machine Check while writing Patch Bits
                               FD  11  546E  6252  BRB 25$   ; Don't allow continue
      06 00000223 9F    02   E0  5470  6253 310$: BBS #2,a#PCS_STATUS_FLAGS,320$
                                5478  6254        ; Check to see if PCS loaded
                       0009  30  5478  6255  BSBW ERROR_CALL  ; Go call the error
                             00  547B  6256 35$: HALT    ; Machine Check while writing PCS
                               FD  11  547C  6257  BRB 35$   ; Don't allow continue
                       0003  30  547E  6258 320$: BSBW ERROR_CALL  ; Go call the error
                             00  5481  6259 45$: HALT    ; Unexpected Machine Check, aborting
                               FD  11  5482  6260  BRB 45$   ; Don't allow continue
```

```
                                5484  6261 ERROR_CALL::
          18 0000FE00 EF    1F    E1  5484  6262  BBC #31,APT_CTRL_FLAGS,10$ ; Branch if not APT mode
             0000FE44 EF    6E    D0  548C  6263  MOVL (SP),APT_ERR_PC  ; Save Error PC
             0000FE40 EF    01    88  5493  6264  BISB2 #1,APT_ERR_FLAG  ; Set Error Flag
                           B7E5   30  549A  6265  BSBW APT_COMMAND
                         00 BE    95  549D  6266  TSTB @(SP)   ; Is next inst a HALT?
                            02    12  54A0  6267  BNEQ 10$   ; Branch if not
                            6E    D6  54A2  6268  INCL (SP)   ; Fudge return over halt
                                  05  54A4  6269 10$: RSB    ; Exit
```

```
                                54A5  6270  .SBTTL Patch bit and micro word area
                                54A5  6271  ; **** THIS IS THE PATCH BIT AND MICRO WORD AREA!!!!!!!!!!!!!!!!! ****
                                54A5  6272  ;
                                54A5  6273  ; The following data structure is where the patch bits and the patch
                                54A5  6274  ; microwords are stored.  This is done by some other utility that will
                                54A5  6275  ; convert the micro2 output file(s) into an ASCII file with the macro32
                                54A5  6276  ; directives necessary to allocate the storage.
                                54A5  6277  ;
                                54A5  6278  ; The first two pages (1024 bytes) are for the Patch Bits.  The 32 pages
                                54A5  6279  ; after that are for the Patch mirowords.
                                54A5  6280  ;
                                54A5  6281
                                54A5  6282 PCS_PTH_BIT_ARRAY::
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  54A5  6283  .LONG  XFFFFFFFF  X26
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  54B5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  54C5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  54D5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  54E5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  54F5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5505
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5515
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5525
                 FFFFFFFF'FFFFFFFF'  5535
                          FFFFDFFF  553D  6284  .LONG  XFFFFDFFF
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5541  6285  .LONG  XFFFFFFFF  X18
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5551
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5561
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5571
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5581
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5591
                          FFFFFFDF  55A1  6286  .LONG  XFFFFFFDF
FFFFFFFF FFFFFFFF'FFFFFFFF'FFFFFFFF'  55A5  6287  .LONG  XFFFFFFFF  X22
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  55B5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  55C5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  55D5
FFFFFFFF FFFFFFFF'FFFFFFFF'FFFFFFFF'  55E5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  55F5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5605
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5615
                 FFFFFFFF'FFFFFFFF'  5625
                          FFFFBFFF  562D  6288  .LONG  XFFFFBFFF
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5631  6289  .LONG  XFFFFFFFF  X5
                          FFFFFFFF'  5641
                          FFFBFFFF  5645  6290  .LONG  XFFFBFFFF
                          FFFFFBFF  5649  6291  .LONG  XFFFFFBFF
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  564D  6292  .LONG  XFFFFFFFF  XB
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  565D
          FFFFFFFF'FFFFFFFF'FFFFFFFF'  566D
                          F7FFFFFF  5679  6293  .LONG  XF7FFFFFF
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  567D  6294  .LONG  XFFFFFFFF  X5
                          FFFFFFFF'  568D
                          DFFFFFFF  5691  6295  .LONG  XDFFFFFFF
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5695  6296  .LONG  XFFFFFFFF  X33
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  56A5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  56B5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  56C5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  56D5
```

```
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  56E5
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  56F5
FFFFFFFF'FFFFFFFF FFFFFFFF'FFFFFFFF'  5705
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5715
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5725
FFFFFFFF'FFFFFFFF'FFFFFFFF FFFFFFFF'  5735
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5745
         FFFFFFFF'FFFFFFFF FFFFFFFF'  5755
                          FBFFFFFF   5761  6297  .LONG  XFBFFFFFF
                          FFFFFFF7   5765  6298  .LONG  XFFFFFFF7
         FFFFFFFF'FFFFFFFF'FFFFFFFF'  5769  6299  .LONG  XFFFFFFFF  X3
                          7FFFFFFF   5775  6300  .LONG  X7FFFFFFF
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5779  6301  .LONG  XFFFFFFFF  X4B
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5789
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5799
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  57A9
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  57B9
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  57C9
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  57D9
FFFFFFFF FFFFFFFF'FFFFFFFF'FFFFFFFF'  57E9
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  57F9
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5809
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5819
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5829
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5839
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5849
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5859
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5869
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5879
FFFFFFFF'FFFFFFFF'FFFFFFFF'FFFFFFFF'  5889
         FFFFFFFF'FFFFFFFF'FFFFFFFF'  5899
                                      58A5  6302 PCS_PTH_MIC_ARRAY::
03000047 28394800 03640300 00472839   58A5  6303   .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364   58B5  6304   .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047   58C5  6305   .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800   58D5  6306   .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300   58E5  6307   .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839   58F5  6308   .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364   5905  6309   .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047   5915  6310   .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800   5925  6311   .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300   5935  6312   .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839   5945  6313   .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364   5955  6314   .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047   5965  6315   .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800   5975  6316   .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300   5985  6317   .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839   5995  6318   .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364   59A5  6319   .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047   59B5  6320   .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800   59C5  6321   .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300   59D5  6322   .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839   59E5  6323   .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364   59F5  6324   .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047   5A05  6325   .OCTA  X03640300004728394800036403000047
00472839 59E5D538 03078470 33554800   5A15  6326   .OCTA  X00472839 59E5D538 03078470 33554800
186BC124 03020470 29464800 03640300   5A25  6327   .OCTA  X186BC124030204702946480003640300
```

```
03000047 28394800 03640300 00472839  5A35  6328  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5A45  6329  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5A55  6330  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5A65  6331  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5A75  6332  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5A85  6333  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5A95  6334  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5AA5  6335  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5AB5  6336  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5AC5  6337  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5AD5  6338  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5AE5  6339  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5AF5  6340  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5B05  6341  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5B15  6342  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5B25  6343  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5B35  6344  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5B45  6345  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5B55  6346  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5B65  6347  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5B75  6348  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5B85  6349  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5B95  6350  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5BA5  6351  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5BB5  6352  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5BC5  6353  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5BD5  6354  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5BE5  6355  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5BF5  6356  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5C05  6357  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5C15  6358  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5C25  6359  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5C35  6360  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5C45  6361  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5C55  6362  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5C65  6363  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5C75  6364  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5C85  6365  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5C95  6366  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5CA5  6367  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5CB5  6368  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472639 48000364  5CC5  6369  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5CD5  6370  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5CE5  6371  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5CF5  6372  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5D05  6373  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5D15  6374  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5D25  6375  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5D35  6376  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5D45  6377  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5D55  6378  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5D65  6379  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5D75  6380  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5D85  6381  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5D95  6382  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5DA5  6383  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5DB5  6384  .OCTA  X28394800036403000047283948000364
```

```
03640300 00472839 48000364 03000047  5DC5  6385  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5DD5  6386  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5DE5  6387  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5DF5  6388  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5E05  6389  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5E15  6390  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5E25  6391  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5E35  6392  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5E45  6393  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5E55  6394  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5E65  6395  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5E75  6396  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5E85  6397  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5E95  6398  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5EA5  6399  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5EB5  6400  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5EC5  6401  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5ED5  6402  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5EE5  6403  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5EF5  6404  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5F05  6405  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5F15  6406  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5F25  6407  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5F35  6408  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5F45  6409  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5F55  6410  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5F65  6411  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5F75  6412  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5F85  6413  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5F95  6414  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5FA5  6415  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  5FB5  6416  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  5FC5  6417  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  5FD5  6418  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  5FE5  6419  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  5FF5  6420  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6005  6421  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6015  6422  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6025  6423  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6035  6424  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6045  6425  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6055  6426  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6065  6427  .OCTA  X48000364030000472839480003640300
03140470 170A4604 002403D8 4A703038  6075  6428  .OCTA  X03140470170A4604002403D84A703038
32D34800 03647B00 05D0E944 48440080  6085  6429  .OCTA  X32D3480003647B0005D0E94448440080
03640300 00472839 480003A4 03D84A70  6095  6430  .OCTA  X03640300004728394800003A403D84A70
48703E80 48000364 03000047 28394800  60A5  6431  .OCTA  X48703E804800036403000047283948000
48000364 03000047 2839881A 71100300  60B5  6432  .OCTA  X48000364030000472839881A71100300
03000047 28394800 03640300 00472839  60C5  6433  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  60D5  6434  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  60E5  6435  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  60F5  6436  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6105  6437  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6115  6438  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6125  6439  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6135  6440  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6145  6441  .OCTA  X00472839480003640300004728394800
```

```
48000364 03000047 28394800 03640300  6155  6442  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  6165  6443  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6175  6444  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6185  6445  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6195  6446  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  61A5  6447  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  61B5  6448  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  61C5  6449  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  61D5  6450  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  61E5  6451  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  61F5  6452  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  6205  6453  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6215  6454  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6225  6455  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6235  6456  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6245  6457  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  6255  6458  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6265  6459  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6275  6460  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6285  6461  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6295  6462  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  62A5  6463  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  62B5  6464  .OCTA  X2839480003640300004728394800036
03640308 00472839 48000364 03000047  62C5  6465  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  62D5  6466  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  62E5  6467  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  62F5  6468  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6305  6469  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6315  6470  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6325  6471  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6335  6472  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  6345  6473  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6355  6474  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6365  6475  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6375  6476  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6385  6477  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  6395  6478  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  63A5  6479  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  63B5  6480  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  63C5  6481  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  63D5  6482  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  63E5  6483  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  63F5  6484  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6405  6485  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6415  6486  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6425  6487  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  6435  6488  .OCTA  X0300004728394800036403000472839
16EC4800 03640300 00472839 48000364  6445  6489  .OCTA  X16EC4800036403000047283948000364
C120A303 047096F2 04525924 03040470  6455  6490  .OCTA  XC120A303047096F20452592403040470
00472839 48000364 03000047 28399801  6465  6491  .OCTA  X0047283948000364030000472839980
48000364 03000047 28394800 03640300  6475  6492  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  6485  6493  .OCTA  X0300004728394800036403000472839
96F48E01 59240200 5DB03038 48000364  6495  6494  .OCTA  X96F48E01592402005DB0303848000364
03640308 387032EE 1801C120 A3008470  64A5  6495  .OCTA  X03640308387032EE1801C120A3008470
00472839 48000364 03000470 3339CE40  64B5  6496  .OCTA  X0047283948000364030004703339CE40
48000364 03000047 28394800 03640300  64C5  6497  .OCTA  X4800036403000047283948003640300
03000047 28394800 03640300 00472839  64D5  6498  .OCTA  X0300004728394800036403000472839
```

```
28394800 03640300 00472839 48000364  64E5  6499  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  64F5  6500  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  6505  6501  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6515  6502  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6525  6503  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6535  6504  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  6545  6505  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  6555  6506  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6565  6507  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6575  6508  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6585  6509  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  6595  6510  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  65A5  6511  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  65B5  6512  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  65C5  6513  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  65D5  6514  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  65E5  6515  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  65F5  6516  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6605  6517  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6615  6518  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6625  6519  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  6635  6520  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  6645  6521  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6655  6522  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6665  6523  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6675  6524  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  6685  6525  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  6695  6526  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  66A5  6527  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  66B5  6528  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  66C5  6529  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  66D5  6530  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  66E5  6531  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  66F5  6532  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6705  6533  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6715  6534  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  6725  6535  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  6735  6536  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6745  6537  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6755  6538  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6765  6539  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  6775  6540  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  6785  6541  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6795  6542  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  67A5  6543  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  67B5  6544  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  67C5  6545  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  67D5  6546  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  67E5  6547  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  67F5  6548  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6805  6549  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  6815  6550  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  6825  6551  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6835  6552  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6845  6553  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6855  6554  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  6865  6555  .OCTA  X0364030000472839480003640300047
```

```
00472839 48000364 03000047 28394800  6875  6556  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6885  6557  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6895  6558  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  68A5  6559  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  68B5  6560  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  68C5  6561  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  68D5  6562  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  68E5  6563  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  68F5  6564  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6905  6565  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6915  6566  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6925  6567  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6935  6568  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  6945  6569  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6955  6570  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6965  6571  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6975  6572  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6985  6573  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  6995  6574  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  69A5  6575  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  69B5  6576  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  69C5  6577  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  69D5  6578  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  69E5  6579  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  69F5  6580  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6A05  6581  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6A15  6582  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6A25  6583  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  6A35  6584  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6A45  6585  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6A55  6586  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6A65  6587  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6A75  6588  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  6A85  6589  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6A95  6590  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6AA5  6591  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6AB5  6592  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6AC5  6593  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  6AD5  6594  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6AE5  6595  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6AF5  6596  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6B05  6597  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6B15  6598  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  6B25  6599  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6B35  6600  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6B45  6601  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6B55  6602  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6B65  6603  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  6B75  6604  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6B85  6605  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6B95  6606  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6BA5  6607  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  6BB5  6608  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  6BC5  6609  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  6BD5  6610  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  6BE5  6611  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  6BF5  6612  .OCTA  X4800036403000047283948000364030
```

```
03000047 28394800 03640300 00472839  6C05  6613  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6C15  6614  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6C25  6615  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6C35  6616  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6C45  6617  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6C55  6618  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6C65  6619  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6C75  6620  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6C85  6621  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6C95  6622  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6CA5  6623  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6CB5  6624  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6CC5  6625  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6CD5  6626  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6CE5  6627  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6CF5  6628  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6D05  6629  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6D15  6630  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6D25  6631  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6D35  6632  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6D45  6633  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6D55  6634  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6D65  6635  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6D75  6636  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6D85  6637  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6D95  6638  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6DA5  6639  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6DB5  6640  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6DC5  6641  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6DD5  6642  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6DE5  6643  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6DF5  6644  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6E05  6645  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6E15  6646  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6E25  6647  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6E35  6648  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6E45  6649  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6E55  6650  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6E65  6651  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6E75  6652  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6E85  6653  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6E95  6654  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6EA5  6655  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6EB5  6656  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6EC5  6657  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6ED5  6658  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6EE5  6659  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6EF5  6660  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6F05  6661  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6F15  6662  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6F25  6663  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  6F35  6664  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  6F45  6665  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  6F55  6666  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  6F65  6667  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  6F75  6668  .OCTA  X03000047283948000364030000472839
28B14800 03640300 00472839 48000364  6F85  6669  .OCTA  X28B14800036403000047283948000364
```

```
C9240302 54E0158D 086D03A4 03D80470  6F95  6670  .OCTA  XC924030254E0158D086D03A403D80470
00472839 186CC538 03010470 28AE9812  6FA5  6671  .OCTA  X00472839186CC5380301047028AE9812
D86CC524 03010470 158C4800 03640300  6FB5  6672  .OCTA  XD86CC52403010470158C480003640300
03000047 28394800 03640300 00472839  6FC5  6673  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  6FD5  6674  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  6FE5  6675  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  6FF5  6676  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7005  6677  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7015  6678  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7025  6679  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  7035  6680  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7045  6681  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7055  6682  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7065  6683  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7075  6684  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  7085  6685  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7095  6686  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  70A5  6687  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  70B5  6688  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  70C5  6689  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  70D5  6690  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  70E5  6691  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  70F5  6692  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7105  6693  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7115  6694  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  7125  6695  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7135  6696  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7145  6697  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7155  6698  .OCTA  X0300004728394800036403000472839
15BD186C C5380301 0470284C 48000364  7165  6699  .OCTA  X15BD186CC5380301047028C48000364
59240338 05F0284D 586CC524 03010470  7175  6700  .OCTA  X5924033805F0284D586CC52403010470
04703FC4 48000364 03000420 15BA085B  7185  6701  .OCTA  X04703FC4480003640300042015BA085B
181BC138 0301CBF0 15BB186B C3700B01  7195  6702  .OCTA  X181BC1380301CBF015BB186BC3700B01
03000047 28394800 03640300 00472839  71A5  6703  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  71B5  6704  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  71C5  6705  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  71D5  6706  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  71E5  6707  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  71F5  6708  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7205  6709  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  7215  6710  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7225  6711  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7235  6712  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7245  6713  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7255  6714  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  7265  6715  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7275  6716  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7285  6717  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7295  6718  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  72A5  6719  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  72B5  6720  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  72C5  6721  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  72D5  6722  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  72E5  6723  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  72F5  6724  .OCTA  X2839480003640300004728394800364
03640300 00472839 48000364 03000047  7305  6725  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7315  6726  .OCTA  X0047283948000364030000472839480
```

```
48000364 03000047 28394800 03640300  7325  6727  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  7335  6728  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  7345  6729  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  7355  6730  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  7365  6731  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7375  6732  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  7385  6733  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  7395  6734  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  73A5  6735  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  73B5  6736  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  73C5  6737  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  73D5  6738  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  73E5  6739  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  73F5  6740  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  7405  6741  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7415  6742  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  7425  6743  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  7435  6744  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  7445  6745  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  7455  6746  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7465  6747  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  7475  6748  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  7485  6749  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  7495  6750  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  74A5  6751  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  74B5  6752  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  74C5  6753  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  74D5  6754  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  74E5  6755  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  74F5  6756  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7505  6757  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  7515  6758  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  7525  6759  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  7535  6760  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  7545  6761  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7555  6762  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  7565  6763  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  7575  6764  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  7585  6765  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  7595  6766  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  75A5  6767  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  75B5  6768  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  75C5  6769  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  75D5  6770  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  75E5  6771  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  75F5  6772  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  7605  6773  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  7615  6774  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  7625  6775  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  7635  6776  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7645  6777  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  7655  6778  .OCTA  X0300004728394800036403000047283
28394800 03640300 00472839 48000364  7665  6779  .OCTA  X2839480003640300004728394800036
03640300 00472839 48000364 03000047  7675  6780  .OCTA  X0364030000472839480003640300004
00472839 48000364 03000047 28394800  7685  6781  .OCTA  X0047283948000364030000472839480
48000364 03000047 28394800 03640300  7695  6782  .OCTA  X4800036403000047283948000364030O
03000047 28394800 03640300 00472839  76A5  6783  .OCTA  X0300004728394800036403000047283
```

```
28394800 03640300 00472839 48000364  76B5  6784  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  76C5  6785  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  76D5  6786  .OCTA  X0047283948000364030004728394800
48000364 03000047 28394800 03640300  76E5  6787  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  76F5  6788  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7705  6789  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  7715  6790  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7725  6791  .OCTA  X0047283948000364030004728394800
48000364 03000047 28394800 03640300  7735  6792  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7745  6793  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7755  6794  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  7765  6795  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7775  6796  .OCTA  X0047283948000364030004728394800
48000364 03000047 28394800 03640300  7785  6797  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7795  6798  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  77A5  6799  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  77B5  6800  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  77C5  6801  .OCTA  X0047283948000364030004728394800
48000364 03000047 28394800 03640300  77D5  6802  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  77E5  6803  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  77F5  6804  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  7805  6805  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7815  6806  .OCTA  X0047283948000364030004728394800
48000364 03000047 28394800 03640300  7825  6807  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7835  6808  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7845  6809  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  7855  6810  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7865  6811  .OCTA  X0047283948000364030004728394800
48000364 03000047 28394800 03640300  7875  6812  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7885  6813  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7895  6814  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  78A5  6815  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  78B5  6816  .OCTA  X0047283948000364030004728394800
48000364 03000047 28394800 03640300  78C5  6817  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  78D5  6818  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  78E5  6819  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  78F5  6820  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7905  6821  .OCTA  X0047283948000364030004728394800
48000364 03000047 28394800 03640300  7915  6822  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7925  6823  .OCTA  X0300004728394800036403000472839
28394800 03640300 00472839 48000364  7935  6824  .OCTA  X2839480003640300047283948000364
03640300 00472839 48000364 03000047  7945  6825  .OCTA  X0364030000472839480003640300047
00472839 48000364 03000047 28394800  7955  6826  .OCTA  X0047283948000364030004728394800
48000364 03000047 28394800 03640300  7965  6827  .OCTA  X4800036403000047283948000364030
03000047 28394800 03640300 00472839  7975  6828  .OCTA  X0300004728394800036403000472839
948C4800 03640300 00472839 48000364  7985  6829  .OCTA  X948C4800036403000047283948000364
03640300 047014AC D80BC100 A3010470  7995  6830  .OCTA  X03640300047014ACD80BC100A3010470
64701490 880C5924 6F000470 948E4800  79A5  6831  .OCTA  X64701490880C59246F00047094 8E4800
48000364 03000470 14AC086B 03640300  79B5  6832  .OCTA  X4800036403000470 14AC086B03640300
03D86070 1494D80B C920A705 04709492  79C5  6833  .OCTA  X03D86070 1494D80BC920A70504709492
14AC986C C5240302 04701491 48005BE4  79D5  6834  .OCTA  X14AC986CC5240302047014914 005BE4
C920A310 04709496 D86BC370 03010470  79E5  6835  .OCTA  XC920A3100470 496D86BC37003010470
047014A1 C81A5924 03004870 149A180C  79F5  6836  .OCTA  X047014A1C81A5924030048701 9A180C
D80BC920 A3008470 9498C800 03640300  7A05  6837  .OCTA  XD80BC920A30084709498C80003640300
03000470 54A3885B 59240338 0470149B  7A15  6838  .OCTA  X0300047054A3885B5924033804701 9B
54A30800 03640B00 04703FC0 C8000364  7A25  6839  .OCTA  X54A308000364 B0004703FC0C8000364
C1100302 4A7054A3 881A0010 03D84A70  7A35  6840  .OCTA  XC11003024A7054A3881A001003D84A70
```

```
4870149F 88005BE4 03384A70 149ED81A  7A45  6841  .OCTA  X4870149F88005BE403384A70149ED81A
48000364 03000470 14A0C81A 59240300  7A55  6842  .OCTA  X48000364030004701 4A0C81A59240300
03010470 54A34800 03640F00 04703FC0  7A65  6843  .OCTA  X03010470 54A3480003640F0004703FC0
14A40800 03640B00 0470003F 186CC538  7A75  6844  .OCTA  X14A4080003640B000470003F186CC538
CB700301 E87014A5 48000364 03000570  7A85  6845  .OCTA  XCB700301E87014A548000364030005 70
607014A7 086B0364 03006470 14A69800  7A95  6846  .OCTA  X607014A7086B0364030064 7014A69800
481B5BE4 03D85070 14A89800 CB700305  7AA5  6847  .OCTA  X481B5BE403D8507014A89800CB700305
03D85070 14AA9800 CB700306 E07014A9  7AB5  6848  .OCTA  X03D8507014AA9800CB700306E07014A9
3FC1480B 59240300 607014AB C81B5BE4  7AC5  6849  .OCTA  X3FC1480B5924030060 7014ABC81B5BE4
03640300 647014AD 88000364 0B000470  7AD5  6850  .OCTA  X03640300647014AD88000364 0B000470
00472839 5809C920 A7040470 343AC869  7AE5  6851  .OCTA  X004728395809C920A7040470343AC869
48000364 03000047 28394800 03640300  7AF5  6852  .OCTA  X4800036403000047283948000364 0300
03000047 28394800 03640300 00472839  7B05  6853  .OCTA  X0300004728394800036403000047 2839
28394800 03640300 00472839 48000364  7B15  6854  .OCTA  X2839480003640300004728394800 0364
03640300 00472839 48000364 03000047  7B25  6855  .OCTA  X0364030000472839480003640300 0047
00472839 48000364 03000047 28394800  7B35  6856  .OCTA  X0047283948000364030004728394 800
48000364 03000047 28394800 03640300  7B45  6857  .OCTA  X4800036403000047283948000364 0300
03000047 28394800 03640300 00472839  7B55  6858  .OCTA  X0300004728394800036403000047 2839
28394800 03640300 00472839 48000364  7B65  6859  .OCTA  X2839480003640300004728394800 0364
03640300 00472839 48000364 03000047  7B75  6860  .OCTA  X0364030000472839480003640300 0047
00472839 48000364 03000047 28394800  7B85  6861  .OCTA  X0047283948000364030004728394 800
48000364 03000047 28394800 03640300  7B95  6862  .OCTA  X4800036403000047283948000364 0300
03000047 28394800 03640300 00472839  7BA5  6863  .OCTA  X0300004728394800036403000047 2839
28394800 03640300 00472839 48000364  7BB5  6864  .OCTA  X2839480003640300004728394800 0364
03640300 00472839 48000364 03000047  7BC5  6865  .OCTA  X0364030000472839480003640300 0047
00472839 48000364 03000047 28394800  7BD5  6866  .OCTA  X0047283948000364030004728394 800
48000364 03000047 28394800 03640300  7BE5  6867  .OCTA  X4800036403000047283948000364 0300
03000047 28394800 03640300 00472839  7BF5  6868  .OCTA  X0300004728394800036403000047 2839
28394800 03640300 00472839 48000364  7C05  6869  .OCTA  X2839480003640300004728394800 0364
03640300 00472839 48000364 03000047  7C15  6870  .OCTA  X0364030000472839480003640300 0047
00472839 48000364 03000047 28394800  7C25  6871  .OCTA  X0047283948000364030004728394 800
48000364 03000047 28394800 03640300  7C35  6872  .OCTA  X4800036403000047283948000364 0300
03000047 28394800 0364030C 00472839  7C45  6873  .OCTA  X0300004728394800036403000047 2839
28394800 03640300 00472839 48000364  7C55  6874  .OCTA  X2839480003640300004728394800 0364
03640300 00472839 48000364 03000047  7C65  6875  .OCTA  X0364030000472839480003640300 0047
00472839 48000364 03000047 28394800  7C75  6876  .OCTA  X0047283948000364030004728394 800
48000364 03000047 28394800 03640300  7C85  6877  .OCTA  X4800036403000047283948000364 0300
03000047 28394800 03640300 00472839  7C95  6878  .OCTA  X0300004728394800036403000047 2839
28394800 03640300 00472839 48000364  7CA5  6879  .OCTA  X2839480003640300004728394800 0364
03640300 00472839 48000364 03000047  7CB5  6880  .OCTA  X0364030000472839480003640300 0047
00472839 48000364 03000047 28394800  7CC5  6881  .OCTA  X0047283948000364030004728394 800
48000364 03000047 28394800 03640300  7CD5  6882  .OCTA  X4800036403000047283948000364 0300
03000047 28394800 03640300 00472839  7CE5  6883  .OCTA  X0300004728394800036403000047 2839
28394800 03640300 00472839 48000364  7CF5  6884  .OCTA  X2839480003640300004728394800 0364
03640300 00472839 48000364 03000047  7D05  6885  .OCTA  X0364030000472839480003640300 0047
00472839 48000364 03000047 28394800  7D15  6886  .OCTA  X0047283948000364030004728394 800
48000364 03000047 28394800 03640300  7D25  6887  .OCTA  X4800036403000047283948000364 0300
03000047 28394800 03640300 00472839  7D35  6888  .OCTA  X0300004728394800036403000047 2839
28394800 03640300 00472839 48000364  7D45  6889  .OCTA  X2839480003640300004728394800 0364
03640300 00472839 48000364 03000047  7D55  6890  .OCTA  X0364030000472839480003640300 0047
00472839 48000364 03000047 28394800  7D65  6891  .OCTA  X0047283948000364030004728394 800
48000364 03000047 28394800 03640300  7D75  6892  .OCTA  X4800036403000047283948000364 0300
03000047 28394800 03640300 00472839  7D85  6893  .OCTA  X0300004728394800036403000047 2839
28394800 03640300 00472839 48000364  7D95  6894  .OCTA  X2839480003640300004728394800 0364
03640300 00472839 48000364 03000047  7DA5  6895  .OCTA  X0364030000472839480003640300 0047
00472839 48000364 03000047 28394800  7DB5  6896  .OCTA  X0047283948000364030004728394 800
48000364 03000047 28394800 03640300  7DC5  6897  .OCTA  X4800036403000047283948000364 0300
```

```
03000047 28394800 03640300 00472839  7DD5  6898  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  7DE5  6899  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  7DF5  6900  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  7E05  6901  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  7E15  6902  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  7E25  6903  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  7E35  6904  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  7E45  6905  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  7E55  6906  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  7E65  6907  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  7E75  6908  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  7E85  6909  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  7E95  6910  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  7EA5  6911  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  7EB5  6912  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  7EC5  6913  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  7ED5  6914  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  7EE5  6915  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  7EF5  6916  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  7F05  6917  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  7F15  6918  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  7F25  6919  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  7F35  6920  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  7F45  6921  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  7F55  6922  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  7F65  6923  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  7F75  6924  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  7F85  6925  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  7F95  6926  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  7FA5  6927  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  7FB5  6928  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  7FC5  6929  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  7FD5  6930  .OCTA  X03640300004728394800036403000047
00472839 48000364 0300C047 28394800  7FE5  6931  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  7FF5  6932  .OCTA  X48000364030000472839480003640300
03000047 28394800 03640300 00472839  8005  6933  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  8015  6934  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  8025  6935  .OCTA  X03640300004728394800036403000047
04703031 48000364 03000047 28394800  8035  6936  .OCTA  X04703031480003640300004728394800
48000364 03000047 28391862 DB240331  8045  6937  .OCTA  X48000364030000472839 1862DB240331
03000047 28394800 03640300 00472839  8055  6938  .OCTA  X03000047283948000364030000472839
28394800 03640300 00472839 48000364  8065  6939  .OCTA  X28394800036403000047283948000364
03640300 00472839 48000364 03000047  8075  6940  .OCTA  X03640300004728394800036403000047
00472839 48000364 03000047 28394800  8085  6941  .OCTA  X00472839480003640300004728394800
48000364 03000047 28394800 03640300  8095  6942  .OCTA  X48000364030000472839480003640300
                                     80A5  6943 PCS_PATCH_END::
                                     80A5  6944  .END
```

PROGRAM_NAME		00000224	R	01	T15_S4		0000363A	R	01
PROMPT		00000267	R	01	T15_S5		0000367C	R	01
PTE_MASK	=	03FF8000	G		T15_S6		000036BA	R	01
RD_NXM	=	00000003	G		T15_S7		00003704	R	01
READY	=	00000007	G		T15_S8		00003746	R	01
RET_PC		00000A07	RG	01	T15_S9		00003794	R	01
RXCS	=	00000020	G		T16_HLT		00003A47	R	01
RXDB	=	00000021	G		T16_S1		00003A48	R	01
SBR	=	0000000C	G		T17_HLT		00003AD2	R	01
SCB		00000400	RG	01	T17_S1		00003AD3	R	01
SCB0		00000400	RG	01	T18_HLT		00003B60	R	01
SCB04		00000404	RG	01	T18_S1		00003B61	R	01
SCB20		00000420	RG	01	T19_HLT		00003BF0	R	01
SCB24		00000424	RG	01	T19_S1		00003BF1	R	01
SCB60		00000460	RG	01	T1_HLT		00000DA9	R	01
SCB64		00000464	RG	01	T1_S1		00000DAA	R	01
SCB8		00000408	RG	01	T1_S2		000010A3	R	01
SCBB	=	00000011	G		T1_S3		000010E2	R	01
SECONDARY_REV	=	00000003	G		T1_S4		00001142	R	01
SLR	=	0000000D	G		T1_S5		00001186	R	01
SPEC TB_TEST		00000A3B	RG	01	T20_HLT		00003C80	R	01
SPTE	=	00010300	G		T20_S1		00003C81	R	01
START		00000AB2	R	01	T20_S2		00003D07	R	01
STRT14		000034E4	R	01	T21_HLT		00003DC3	R	01
STRT2		0000123C	R	01	T21_S1		00003DC4	R	01
STRT3		00001328	R	01	T21_S2		00003E4A	R	01
STRT4		00001550	R	01	T22_HLT		00003F0A	R	01
SYNC_POINT	=	00FFF460	G		T22_S1		00003F0B	R	01
SYS_PTBL	=	00010000	G		T23_HLT		00003FCD	R	01
T10_HLT		000021A1	R	01	T23_S1		00003FCE	R	01
T10_INIT1		0000289F	R	01	T24_HLT		00004090	R	01
T10_INIT2		000030A5	R	01	T24_S1		00004091	R	01
T10_S1		000021A2	R	01	T25_HLT		00004151	R	01
T10_S2		000028BA	R	01	T25_S1		00004152	R	01
T10_S3		00003000	R	01	T26_HLT		00004215	R	01
T10_S4		000030C0	R	01	T26_INIT1		000042EE	R	01
T11_HLT		00003189	R	01	T26_S1		00004216	R	01
T11_S1		0000318A	R	01	T26_S2		00004308	R	01
T12_HLT		00003247	R	01	T27_HLT		000043F9	R	01
T12_S1		00003248	R	01	T27_INIT1		000044D2	R	01
T13_HLT		00003321	R	01	T27_S1		000043FA	R	01
T13_S1		00003322	R	01	T27_S2		000044EC	R	01
T14_HLT		000034D6	R	01	T28_HLT		000045DD	R	01
T14_S1		000034D7	R	01	T28_INIT1		000046B8	R	01
T15_HLT		00003576	R	01	T28_S1		000045DE	R	01
T15_S1		00003577	R	01	T28_S2		000046D5	R	01
T15_S10		000037E2	R	01	T29_HLT		000047CA	R	01
T15_S11		00003824	R	01	T29_INIT1		000048A5	R	01
T15_S12		00003862	R	01	T29_S1		000047CB	R	01
T15_S13		000038AC	R	01	T29_S2		000048C2	R	01
T15_S14		000038EE	R	01	T2_HLT		0000122D	R	01
T15_S15		0000393C	R	01	T2_S1		0000122E	R	01
T15_S16		0000398A	R	01	T30_HLT		000049B7	R	01
T15_S17		000039CC	R	01	T30_INIT1		00004A50	R	01
T15_S2		000035AA	R	01	T30_S1		000049B8	R	01
T15_S2A		000035DA	R	01	T30_S2		00004A53	R	01
T15_S3		0000360A	R	01	T31_HLT		00004B23	R	01

```
T31_INIT1                       00004BBC R      01        TB_PAT_C                        00000318 RG     01
T31_S1                          00004B24 R      01        TB_PAT_D                        0000033C RG     01
T31_S2                          00004BBF R      01        TB_VAL                        = 0000001F  G
T32_HLT                         00004C90 R      01        TCB                             00000600 RG     01
T32_S1                          00004C91 R      01        TCB0                            00000600 RG     01
T32_S1_START                    00004CB3 R      01        TCB10                           00000610 RG     01
T32_S1_X                        00004D66 R      01        TCB100                          00000700 RG     01
T33_HLT                         00004DFA R      01        TCB14                           00000614 RG     01
T33_S1                          00004DFB R      01        TCB18                           00000618 RG     01
T33_S2                          00004E4D R      01        TCB1C                           0000061C RG     01
T33_S3                          00004EBA R      01        TCB20                           00000620 RG     01
T33_S4                          00004F2A R      01        TCB24                           00000624 RG     01
T34_HLT                         00004FD6 R      01        TCB28                           00000628 RG     01
T34_S1                          00004FD7 R      01        TCB2C                           0000062C RG     01
T35_HLT                         0000505F R      01        TCB30                           00000630 RG     01
T35_S1                          00005060 R      01        TCB34                           00000634 RG     01
T35_S2                          0000512D R      01        TCB38                           00000638 RG     01
T35_S3                          000051EB R      01        TCB3C                           0000063C RG     01
T35_S4                          000052B4 R      01        TCB4                            00000604 RG     01
T35_S5                          00005319 R      01        TCB40                           00000640 RG     01
T3_HLT                          00001319 R      01        TCB44                           00000644 RG     01
T3_S1                           0000131A R      01        TCB48                           00000648 RG     01
T3_S2                           0000139C R      01        TCB4C                           0000064C RG     01
T3_S3                           00001416 R      01        TCB50                           00000650 RG     01
T3_S4                           00001490 R      01        TCB8                            00000608 RG     01
T4_HLT                          00001543 R      01        TCB84                           00000684 RG     01
T4_S1                           00001544 R      01        TCB88                           00000688 RG     01
T4_S2                           0000159A R      01        TCB8C                           0000068C RG     01
T4_S3                           000015EA R      01        TCB90                           00000690 RG     01
T4_S4                           0000163B R      01        TCB94                           00000694 RG     01
T4_S5                           0000168C R      01        TCB98                           00000698 RG     01
T4_S6                           000016DD R      01        TCB9C                           0000069C RG     01
T5_HLT                          00001767 R      01        TCBA0                           000006A0 RG     01
T5_S1                           00001768 R      01        TCBA4                           000006A4 RG     01
T5_S2                           00001891 R      01        TCBA8                           000006A8 RG     01
T6_HLT                          00001A2A R      01        TCBAC                           000006AC RG     01
T6_S1                           00001A2B R      01        TCBB0                           000006B0 RG     01
T6_S2                           00001ADD R      01        TCBB4                           000006B4 RG     01
T7_HLT                          00001B98 R      01        TCBB8                           000006B8 RG     01
T7_S1                           00001B99 R      01        TCBBC                           000006BC RG     01
T8_HLT                          00001C52 R      01        TCBC                            0000060C RG     01
T8_S1                           00001C53 R      01        TCBC0                           000006C0 RG     01
T8_S2                           00001CC4 R      01        TCBC4                           000006C4 RG     01
T9_HLT                          00001D6A R      01        TCBF8                           000006F8 RG     01
T9_INIT1                        00001E22 R      01        TCBFC                           000006FC RG     01
T9_INIT2                        000020B1 R      01        TEMP                          = 0000032B R      01
T9_S1                           00001D6B R      01        TEST_1                          00000D70 RG     01
T9_S2                           00001E40 R      01        TEST_10                         00002168 RG     01
T9_S3                           00002000 R      01        TEST_11                         00003150 RG     01
T9_S4                           000020CA R      01        TEST_12                         0000320E RG     01
TBDR                          - 00000024  G               TEST_13                         000032E8 RG     01
TBIA                          = 00000039  G               TEST_14                         0000349D RG     01
TBIS                          = 0000003A  G               TEST_15                         0000353D RG     01
TB_DATA                       = 0000003B  G               TEST_16                         00003A0E RG     01
TB_ER_BIT                     = 00000002  G               TEST_17                         00003A99 RG     01
TB_PAT_A                        000002F0 RG     01        TEST_18                         00003B27 RG     01
TB_PAT_B                        00000304 RG     01        TEST_19                         00003BB7 RG     01
```

```
TEST_2                          000011F4 RG     01
TEST_20                         00003C47 RG     01
TEST_21                         00003D8A RG     01
TEST_22                         00003ED1 RG     01
TEST_23                         00003F94 RG     01
TEST_24                         00004057 RG     01
TEST_25                         00004118 RG     01
TEST_26                         000041DC RG     01
TEST_27                         000043C0 RG     01
TEST_28                         000045A4 RG     01
TEST_29                         00004791 RG     01
TEST_3                          000012E0 RG     01
TEST_30                         0000497E RG     01
TEST_31                         00004AEA RG     01
TEST_32                         00004C57 RG     01
TEST_33                         00004DC1 RG     01
TEST_34                         00004F9D RG     01
TEST_35                         00005026 RG     01
TEST_4                          0000150A RG     01
TEST_5                          0000172E RG     01
TEST_6                          000019F1 RG     01
TEST_7                          00001B5F RG     01
TEST_8                          00001C19 RG     01
TEST_9                          00001D31 RG     01
TODR                          = 0000001D
TOP_OF_STACK                    00000A00 RG     01
TXCS                          = 00000022  G
TXDB                          = 00000023  G
TYPE_END_PASS                   00000D19 R      01
TYPE_NO_PCS                     00000D56 R      01
TYPE_PROMPT                     00000D3C R      01
UET_ADDR                      = 00FFF460  G
UET_CNTL                      - 00FFF464  G
UNI_MAP                       = 00F30800  G
UNI_PFN                         000002DC RG     01
VA1                             00000348 RG     01
VA2                             0000034C RG     01
VA3                             00000350 RG     01
WT_NXM_SRV                      00000BCC R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
CACHTB	000080A5 (32933.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	44	00:00:00.10	00:00:00.89
Command processing	114	00:00:00.33	00:00:01.51

Pass 1	1345	00:00:25.55	00:01:48.71
Symbol table sort	14	00:00:01.10	00:00:07.50
Pass 2	2123	00:00:16.34	00:01:59.36
Symbol table output	62	00:00:00.31	00:00:00.97
Psect synopsis output	5	00:00:00.03	00:00:00.12
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	3710	00:00:43.77	00:03:59.11

The working set limit was 500 pages.
188193 bytes (368 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 381 non-local and 517 local symbols.
6944 source lines were read in Pass 1, producing 78 object records in Pass 2.
2 pages of virtual memory were used to define 1 macro.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
SYS$SYSROOT: SYSLIB STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/LIS ECKAL